The Bartlett School of Graduate Studies - University College London

Parasitic Ecologies:

extending space through diffusion-limited aggregation models

Georgia Kachri

This dissertation is submitted in partial fulfilment of the requirements for the degree of Master of Science in Adaptive Architecture & Computation from University College London

University of London, September 2009

I, Georgia Kachri, confirm that the work presented in this thesis is my own.
Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Georgia Kachri

Abstract

Parasitic architecture allows the creation of flexible structures that feed off existing infrastructure offering solid answers to the problem of structural density of cities and the need of temporary accommodations. Additionally, modular systems potentially provide forms with great complexity out of simplicity.

This thesis investigates the evolution of self-sustained parasitic structures that evolve by creating aggregation forms well adapted to their hosts. The growth process of the parasite is inspired by the fungal colonies and is based on the rules of the diffusion-limited aggregation extended to support force analysis and maintain structural stability. The hypothesis is that the modular development of forms, capable to adapt to the built environment without affecting the stability of the existing infrastructure, provides easy and low-cost solutions for extending and rejuvenating architectural space. Moreover, the natural growth process of diffusion-limited aggregation allows an endless evolution of additional spaces within a city in respect to the character of its landscape and human's current needs. The final findings propose new ways of space perception and introduce a strategy for exploiting space.

Word count: 9975

Acknowledgements

First I would like to thank my supervisor Alasdair Turner for his constant guidance, motivation and encouragement, Sean Hanna for the inspirational discussions and Ruairi Glynn for the alternative directions of investigation.

I would also like to thank my family and especially my father Nikos and my mother Maria for their endless care and unconditional support all these years.

Last but not least, special thanks to my partner Tasos Varoudis for his invaluable computational and architectural advice, encouragement and patience until the end.

Table of Contents

List of Figures

1. Introduction

"Study nature, love nature, stay close to nature. It will never fail you."
Frank Lloyd Wright

In most cases, buildings are massive, permanent and static providing an inability to deal with the amount of energy embodied in their structure. Yet the 20th century has seen the emergence of structures that use the model of nature as the generating force for their form. Likewise living organisms that grow under specific environmental conditions and adapt to their context, these architectural models interact and evolve in harmony with natural forces. This approach, which demands combined effort of biology and computer science, focuses on the generative process of the structure and ensures the creation of forms balanced with their natural context. Drawing on ecological relations found in nature, this thesis is inspired from the adaptive strategies of fungal parasitic colonies to examine how the implementation of biological analogies in relation to architectural design can support the emergence of self-organized structures.

1.1 Towards biological analogies

The impact of biological models on architecture has been studied extensively (Alexander, 1964; Steadman, 2008; Thomson, 1942; Ball, 1999) and has proposed new ways for interfering with design. The basis can be found not on the imitation of the evolution of species collectively but on the process of their growth. As the growth of an organism follows a fixed evolutionary process in response to environmental conditions, design must be informed constantly from these constraints that affect its progress.

Thomson (1942) was one of the first researchers, who addressed, the interrelation between growth and form. In his study, the form of organisms is considered as an event in space over time rather than a spatial configuration, which means that the growth process is the basis for the resulted form. Drawing on the same discipline,

more recently, Ball (1999) explored the pattern formation in nature suggesting that all the forms that appear can be considered as results of specific growth processes. Therefore, the simulation of a specific process can lead to the generation of the desired form. As Steadman (2008) argues in an attempt of introducing growth process into architectural design, during the development there should be a frequent interaction between the growing design and its environment. Steadman's assumption can be considered as derivation from Lamarck's theory of adaptation, in which an organism-system grows in response to environmental forces but also influences its environment correspondingly (Gould, 2002).

Studying and interpreting biological analogies in relation to architectural design allows the development of emergent and self-organized structures. Emergent structures are the complex systems that arise from relatively simple interactions while self-organization refers to the ability of a system to adapt in a continually changing environment (Camazine et al., 2003).

In a world, where energy supplies become limited and the overbuilt environment seeks to accommodate new functionalities, the exploration of ecological relationships and evolutionary mechanisms is more than crucial for architecture.

1.2. Parasitism - An ecological relation

In biology, the term parasitism refers to an ecological relationship (Bush et al., 2001) between two organisms, where one organism, the parasite, takes advantage from its host. Parasites evolve in response to defense mechanisms of their hosts and exhibit a higher reproductive potential than their hosts. In fact, half of all species of organisms are parasites creating a continuum of interactions between species that negotiates natural equilibrium. There are several categorizations of parasites regarding their relation to the hosts. Most importantly parasites can be either endoparasites, which live inside their hosts, or ectoparasites that live on but not within their hosts (Bush et al. 2001; Deverall, 1981). Parasites have significant impact on ecosystem functioning by regulating populations of dominant species,

applying top-down control on populations from lower trophic levels, shifting from parasitic to mutualistic[1] interactions with their hosts and transmitting nutrients to more or less efficient recycling pathways (Thomas et al., 2005). Among the great diversity of parasitic species, a class that presents significant importance is this of fungi.

Fungi represent one of the three evolutionary branches of multicellular organisms, as they are equivalent to plants and animals. However, their dominance is based not only on their great amount but also to their usefulness in the cycle of life, as they are the main decomposers and recycles of organic matter. Fungi are remarkably tolerant of external factors such as dryness or high and low temperature and thus they can spread everywhere in terms of space (Jennings and Lysek, 1996). Fungi are modular organisms (Carrol and Wicklow, 1992), which means that they grow by repeated iterations of parts called modules. In a fungal colony the module is a thin, tubular structure called hyphae that extends and presents apical characteristics of growth (Moore, 1998). Fungal modular growth (Harper et al., 1986) leads to root-like forms (Gow and Gadd, 1995) with great plasticity, complexity and diversity. Although, their growth mechanisms are of primitive intelligence, they allow for the creation of complicated networks that can be considered as three-dimensional lattices (fig. 1).

Figure 1: Fungal colonies

[1] Mutualism is a biological interaction between two organisms, where each individual derives a fitness benefit.

Fungal morphogenesis and parasitic behaviour provide potentials for applying natural rules into manmade environment. The placement of structures, which on the one hand, feed from existing infrastructure and on the other hand, grow in respect to natural affordances, can be an answer to the recurrent problem of dense structured cities.

1.3. Aim and Objectives

The aim of this thesis is to address the generation of parasitic structures, which evolve by creating aggregation forms well adapted to their hosts. The parasites are considered as embedded networks to existing infrastructure so that their evolution can take place even above the ground to roofs, facades etc. and introduce possibilities of spaces above and below the built environment. The adaptation of a new generated system into the existing built environment allows the examination of extending the space of existing buildings and generating new patterns to available paths. Furthermore, parasitic systems have many strategies for surviving. There are methods of attachment, reproduction, and sustaining themselves to hosts. To develop a new space within or upon an existing structure requires a strategy that not only provides a new program element, but a method to maintain an attachment and structural integrity.

The main objective is to simulate the growth of a parasite in a three-dimensional model in respect to some of the rules that are applied to fungal colonies. The growth process will follow the rules of Diffusion Limited Aggregation method and some of its extensions (see section 2.3.). Diffusion-limited aggregation algorithm provides a basis for understanding a wide range of natural pattern formation phenomena and fractal growth processes like the evolution of fungal colonies. However, fungal colonies act as a starting point for understanding self-organized and adaptive systems that present a parasitic behaviour and thus the methodological approach will not focus on precise replications of the fungal growth.

The modular growth of fungi allows for the creation of a three-dimensional lattice. For this particular reason, the module, which replicates the fungal hyphae, is expressed with the use of a truncated octahedron. This space filling polyhedron allows for space tessellation and also exhibits great structural stability. Subsequently, apart from the simulation of the growth process, an analysis on the overall stability of the structure will be implemented. The calculation of the principal forces that are applied to the system is essential not only for ensuring a convenient self-organized and self-sustained structure but also for providing an architectural solution able to be assembled.

The hypothesis of this research is that by using diffusion-limited aggregation (DLA) to develop architectural forms with parasitic behaviour, the functional space within cities will potentially extend and the inefficient open spaces will be rejuvenated. The process of DLA allows for modifications without transforming remarkably the character of the city's landscape and the analogies between built and open spaces. The ability of the parasites to maintain structural stability will provide easy and low-cost constructive mechanisms to be applied and self-sustained structures to emerge. Additionally, the evolutionary process, which results from the use of DLA algorithm, can also proceed by human's intervention. For instance, people can add or remove modules from the aggregate in respect to their current needs. Finally, the resulted forms are expected to expand architectural possibilities by introducing spaces with fractal dimensionality.

Specifically, the project will contain case studies for implementing aggregation models to different conditions such as dense built environments and open spaces. The models will follow a modular growth similar to the fungal growth in order to generate a three-dimensional lattice. The simulation of the growth process as well as the structural analysis will be implemented through applications developed in Java programming language.

1.4. Structure of the thesis

The second chapter reviews the relevant literature on completed works as a summary to the objectives of this research. The literature is divided into three parts from which the first presents completed projects on parasitic architecture, the second reviews aggregation models and space filling geometry, and the third refers to fractal growth simulations in order to cover the methodological perspective. The third chapter describes the methodology that has been carried out for this particular research and explains the applied algorithms. Chapter 4 presents the results through specific experiments on the basic characteristics of the structure. Chapter 5 examines the adaption of the resulted structure to different built environments. The sixth chapter consists of a critical evaluation of the results and a discussion for further experimentation. The final chapter provides a conclusion of the research by highlighting the main subjects and its importance for architectural practices.

2. Literature Review

This section presents selected projects and theories with critical thinking in order to cover all different aspects of this research. The first section presents parasitic structures while the second provides information on the character of aggregation models. The final section exhibits theories on fractal growth phenomena and especially on diffusion-limited aggregation to cover the methodology that is going to be undertaken.

2.1. Parasitic Architecture

The term Parasitic Architecture refers to flexible and temporary structures that feed off existing infrastructure (Allen et al., 2003). This idea has received considerable attention from architects, who envisioned architecture not as a permanent space for humans to live in, but as a motivation for social engagement, freedom and endless transportation. Constant Nieuwenhuys was one of the first architects that addressed the mobility of architecture (Wigley, 1998). By designing, the New Babylon in 1959-74, he proposed a nomadic utopian city that would place its center of attention on people's activities and would grow in respect to this behaviour forming accessible to everyone networks (fig. 2). The basic elements of the network can be described as autonomous modules that intercommunicate assembling a continuous endless space.

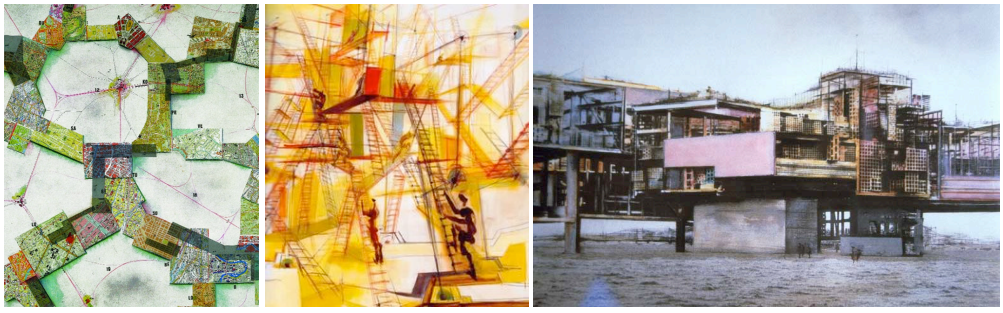


Figure 2: New Babylon by Constant (source: www.arpla.fr/canal2/figureblog/?cat=26)

Drawing on the same discipline, Yona Friedman in 1958 proposed the Spatial City, which consists of mobile, temporary and lightweight structures rather than the rigid and inflexible structures of traditional architecture (Jencks, 1987). He suggested a frame-grid as the infrastructure on which users can build and occupy their own space. Inspired by the work of Friedman, Archigram (www.archigram.net) in 1964 proposed the Walking City, in which mobile robotic structures with their own intelligence could move to wherever was needed. Even though, these concepts appear more than utopian to evolve and embed into current culture, some of their underlying ideas if applied, would exhibit great interest.

From another perspective, parasitic structures can be considered as self-contained new buildings that are attached to an existing structure. The attached buildings take advantage of their hosts regarding energy and water supplies and also structural stability in order to extend space. An example on this category is the La Palmas Parasite designed by Kortknie Stuhlmacher Architects in 2001 (www.kortekniestuhlmacher.nl). The parasite is a prototype for a new form of urban housing that explores the relationship between sustainability and prefabrication (fig. 3). Its prefabricated panels are manufactured from waste wood and need four days for assembling. The La Palmas Parasite is designed to take advantage of its host water and heating systems and is supported by the walls of the existing building. The structural dependence of the parasite is an important limitation for architectural constructions. Intuitively, adding a set of new forces to existing infrastructure is not always the best solution. The ability of a parasitic structure to be at some possible point self-sustained could provide a better answer on how this parasite adapts in a specific built environment.

A current example, which combines both of the above approaches, is this of Micro Dwellings (fig. 4), an idea developed by N55 architects (http://www.n55.dk/). Micro Dwellings consists of equal sized modules with space filling geometry. In fact, these modules are movable and can be placed onto existing buildings or even under water. However, this idea does not exhibit any interest on the growth process of the modules and does not present any answer on how these modules evolve as self-organized structures that at some point reach equilibrium.

Figure 3: The La Palmas Parasite by Kortknie Stuhlmacher Architects
(source: www.kortekniestuhlmacher.nl)

Figure 4: Micro Dwellings by N55 (source: http://www.n55.dk/)

Considering the over-built environment, it is obvious that there is a lack of space for constructing new buildings to accommodate human needs. Parasitic architecture can provide solid answers to both structural density and temporary accommodations. On the one hand, parasites can take advantage of inactive spaces and rejuvenate urban life without the need of empty space and on the other hand because of their flexibility can grow in respect to the new nomadic way of living.

2.2. Aggregation Models

As presented in the introduction, this research explores the evolution of parasitic aggregation models through the DLA method. Currently, there is a lack of implementations in the field of diffusion-limited aggregation models in architectural design. Therefore, this section presents some examples of implemented aggregation models to provide information on the different possibilities that an aggregation model can offer in architectural design.

The process of effectively repeated elements, modules, to generate a unique whole is an idea that has influenced the work of several architects providing forms with great interest and complexity out of simplicity. A well-known example is the Blobwall (fig. 5) by Lynn (http://www.glform.com/blobwall.html). In Blobwall, an initial module is replicated to form wall-type patterns, in which each module overlaps with its neighbours. The module is a robotically cut mass-produced shape with low-density that is assembled through interlocking precision to form the wall. Nevertheless, the interior space that is embedded between a set of walls does not result as a continuum of the generative process. Instead, Blobwall acts only as a boundary that separates a space and its context.

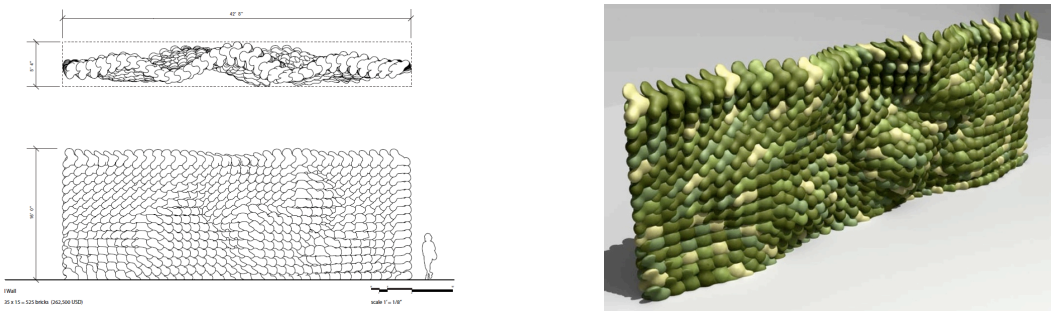


Figure 5: Blobwall by Greg Lynn (source: http://www.glform.com/blobwall.html)

Aranda and Lasch (2006) in their project Grotto Pavilion (fig. 6) implemented an algorithm for assembling prototypes aperiodically. The solution is based on both Danzer tiling and voronoi geometries. The pavilion consists of four modules that fit together in a variety of ways to form a three-dimensional pattern. Grotto Pavilion does not present a solution about its evolution through time. Once the pavilion is completed, the generative process stops.

Moreover, the water cube (National Swimming Center) that was designed for the Beijing Olympics by PTW Architects is an example of space-filling geometry (fig. 7a, 7b). The structure of the water cube is based on the Weaire-Phelan model (fig. 7c) that uses two kinds of cells of equal volume, an irregular pentagonal dodecahedron and a tetrakaidecahedron with two hexagons and twelve pentagons. However, it is not based on a growth process that allows the form to evolve continually from the

interior to exterior as the resulted foam acts only as a ‘shell’ to the interior cubic space.

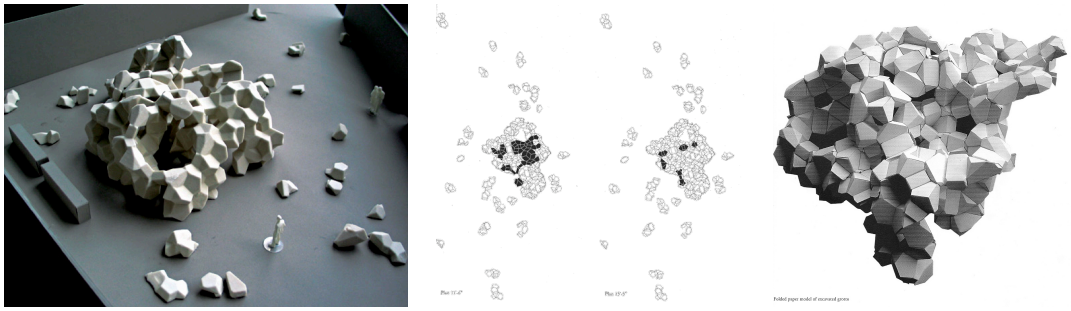


Figure 6: Grotto Pavilion by Aranda and Lasch (source: Aranda & Lasch, 2006)

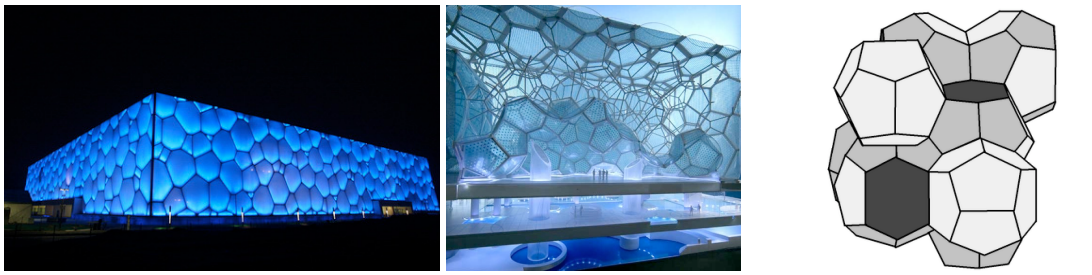


Figure 7: a,b: Water Cube by PTW Architects (source: http://www.ptw.com.au/)

c: Weaire-Phelan structure

2.3. Growing fractal forms

Aggregation models, which produce complicated geometrical objects, represent a recent approach to the problem of pattern formation. A wide category of growing patterns is characterized by an open branching structure that is described as fractal geometry. In fact, fractals are complex shapes that can only be expressed by a non-integer (fractal) dimensionality and are applicable as mathematical models for many objects in nature (Mandelbrot, 1983). For growing fractals, the fractal dimension D can be defined through the linear function $N(L) \sim L^D$, where L is the linear size of the object and N(L) is the number of the balls or boxes of unit volume l = 1 that are needed to cover the structure (Vicsek, 1992). The main characteristic of fractals is

that they are self-similar, which means that if someone cuts out a part of them and enlarge it, the resulting object will appear to be similar to the prototype.

Aristid Lindenmayer (1968) developed a system to simulate the growth process of branching structures and fractal-like forms. His model, known as L-system is based on rules that are applied iteratively starting from an initial state. At first, L-systems were designed for providing a formal description of the development of simple multicellular organisms, and demonstrating relationships between adjacent plant cells. Later, L-systems were extended for simulating more complex branching structures. Nevertheless, this model does not allow for the introduction of geometric restrictions in the growth of a non-deterministic process that is continuously influenced by the environment (Kaandorp, 1994). For instance, as shown below (fig. 8), objects intersect with one another. This demands for additional decisions to be made during the growth process such as removing, overlapping or connecting branches, which makes the system less capable for growing geometric models where restrictions are necessary.

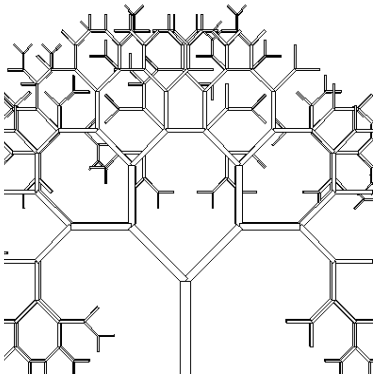


Figure 8: L-System tree showing branches intersection

Several natural growth processes are influenced by non-local settings like temperature and so distant points in addition to the neighborhood behavior govern the system. To examine these far from equilibrium phenomena, in which diffusing particles are added to a growing aggregate, can be studied by approaches based on diffusion-limited aggregation (DLA). The DLA was proposed by T.A. Witten and L.M.

Sander (1981) in order to simulate the formation of clusters by particles diffusing through a medium that collides the particles with each other as they move. DLA through a simple implementation provides a wealth of behaviour and a basis for interpreting a large range of natural pattern formation phenomena. There is a wealth of research on diffusion-limited aggregation usually in two dimensions on models of fractal growth processes such as river networks, plant branching, frost on glass, electro-deposition, lightning, mineral deposits, and coral.

The growth rule is remarkably simple. First, an immobile seed is located on the plane. A walker is then launched from a random position far away and is allowed to diffuse (walk at random). If it touches the seed, it is immobilized instantly and becomes part of the aggregate. Then similar walkers are launched one-by-one and each of them stops upon hitting the cluster. After launching a few hundred particles, a cluster with intricate branch structures results. The resulted fractals have a randomly branching open structure and look stochastically self-similar (fig.9).

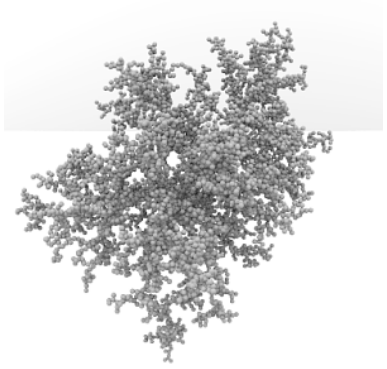


Figure 9: 3D diffusion-limited aggregation model (source: http://markjstock.org/)

There are several growth models that are presented as extensions of the DLA method. Diffusion Limited Desposition (DLD) (Meakin, 1983; Rácz and Vicsek, 1983) represents a similar to DLA case. The main difference is that instead of a single seed, there is a d_s dimensional surface on which diffusing particles are attached. The presence of the surface allows for more complicated systems to emerge. The

simulation below (fig. 10) exhibits an example of DLD, in which a forest of clusters is the result of the growth on a surface.

Another extension of DLA is the ballistic aggregation (Ramanlal and Sander, 1985), models. During ballistic aggregation the particles move along straight lines until they encounter the aggregate and stick to it (fig. 11). When in ballistic aggregation the single seed is replaced from a surface, the model that results is known as ballistic deposition (Meakin et al., 1986). In this case, the particles are released from randomly chosen launching points and move along parallel straight lines until they are attached to the aggregate (fig. 12).

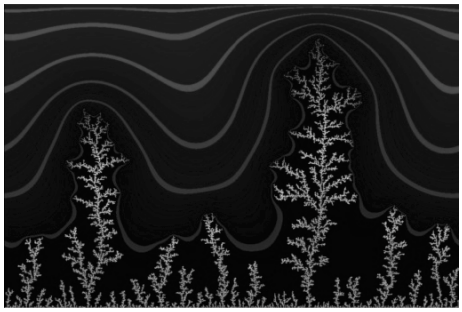


Figure 10: 2D diffusion-limited deposition model
(source: http://algorithmicbotany.org/)

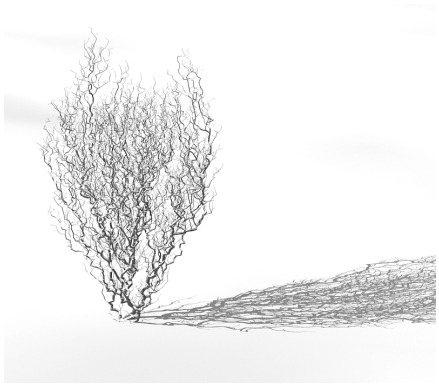


Figure 11: 3D ballistic aggregation model
(source: http://markjstock.org/)

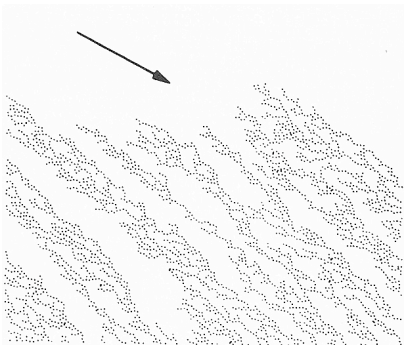


Figure 12: 2D ballistic deposition model
(source: Vicsek, 1992)

Even though there is a wealth of simulations based on the DLA method in the fields of biology and physics, there are only a few implementations of the algorithm in the

field of architectural design. A very interesting approach is the work of David Sutton on the generation of a series of vessels (fig. 13). The growth of the three-dimensional structures, which is based on the implementation of DLA algorithm, is contained within a set of virtual surfaces that define the resulted form. The simulation model of root system was brought into reality through rapid manufacturing processes (http://local.wasp.uwa.edu.au/~pbourke/fractals/dla3d/). DLA application for aggregation models is a rather unexplored field in architecture. Consequently, this offers a great opportunity for new emerging forms to be explored.

Figure 13: DLA vessels by David Sutton (source: http://www.detnk.com/node/167)

3. Methodology

As discussed above, the hypothesis of this research is that the use of DLA to develop self-sustained modular structures would provide the potential for extending architectural space in respect to the character of the existing infrastructure.

The basis of the presented thesis is a computational three-dimensional simulation of a diffusion-limited aggregation model written in Java programming language. Additionally, diffusion-limited deposition, ballistic aggregation and ballistic deposition are also studied. Moreover, in order to ensure the development of a self-sustained structure, a calculation of the main forces that are created within the system was undertaken.

The methodological approach is actually divided into three phases. The first phase includes experimentation on diffusion-limited aggregation models that consist of particles connected in a linear dendritic path. This implementation was essential for experimenting with the less probable rules and understanding the basic principles of DLA. The second section focuses on a force analysis of the system as a starting point for later use. The third section introduces space-filling geometry as a replacement to particles. The objective is to provide the necessary constraints in order to attach the polyhedra with aligned shared faces within the aggregate. Additionally, a final analysis on the stability of the system and also a development of a number of constraints related to the surrounding built environment (see section 5) are implemented.

3.1. Aggregation Model

3.1.1. Experimenting on DLA

At the very beginning, the model was able to form three-dimensional dendritic structures by connecting the aggregated particles with lines. The first implementation was written in openFrameworks framework, which uses a

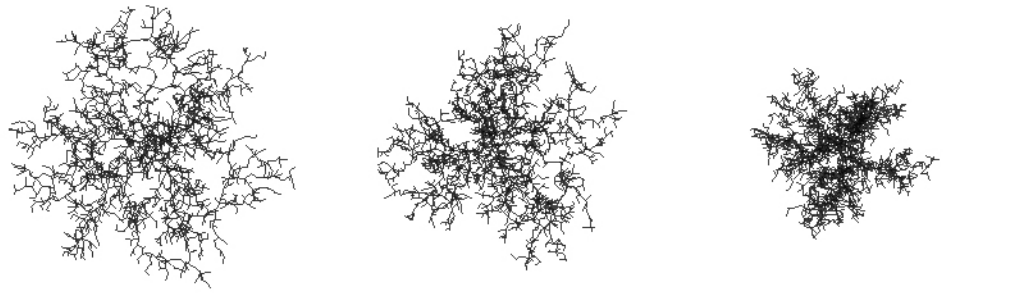
combination of C++ and OpenGL as a core, and was based on a two-dimensional simulation of DLA written on Java/Processing programming language by Turner (http://www.aac.bartlett.ucl.ac.uk/processing/samples/dla/); a two-dimensional implementation by Bourke, written in C programming language, as well as his work on DLA in general (http://local.wasp.uwa.edu.au/~pbourke/fractals/dla/); and a three-dimesional Java/Processing implementation (http://www.openprocessing.org/visuals/?visualID=2044). The use of openFrameworks was important for two basic reasons. First, because it is quicker allows for easier experimentation and second it was better for understanding the algorithm as C++ programming language demands for more analytical and clear manipulation of the program's settings.

The algorithm was set due to the rules of DLA (see 2.3.) with an additional stickiness probability, which would allow in specified degree particles to attach the aggregate. The algorithm is described below:

Algorithm for Diffusion Limited Aggregation with stickiness probability:

1. **SET** the radius of the particle system PR.
2. **SET** the stickiness probability SP.
3. **CREATE** a 3D vector at the center of the system CVEC.
5. **CREATE** an Array List of vectors to contain the aggregate particles AL.
6. **CREATE** a walker W.
7. **SET** the radius of the sphere RS, in which the walker will search for aggregate particles.
8. **CREATE** the initial seed particle IS at the center CVEC of the system.
9. **ADD** IS to AL.
9. **SET** the magnitude of W equal to RS.
10. randomize the direction of W.
    a. **IF** the distance between W and CVEC is more than RS + PR, the walker is out of the sphere
       THEN W = W*(-1).
11. **FOR** each item J in the list AL.
    a. **SET** a random number RN.
    b. **IF** the distance between W and J is less than PR **AND** SP is more than RN.
       THEN add a new particle P to AL.
       i. **IF** the distance D between W and CVEC is more than RS – PR.
          THEN RS = D + PR.
       ii. randomize the direction of W.
    c. **ELSE** continue searching until finding an aggregate particle in the list AL.
12. **CONTINUE** steps from 9 to 12 until getting an aggregation of the desirable size AL = S.

After an aggregation of 3000 particles (fig. 14), it became obvious that the less the sticking probability was the more dense became the structure.



a) stickiness probability 1.0 b) stickiness probability 0.5 c) stickiness probability 0.1

Figure 14: 3D DLA simulation with stickiness probability

When specify the direction of the walker rather than set it to random values, the result is thoroughly different. Figure 15 shows an aggregate of 1000 Particles with a walker set at 30° degrees. However, the stickiness probability still affects the density of the structure.

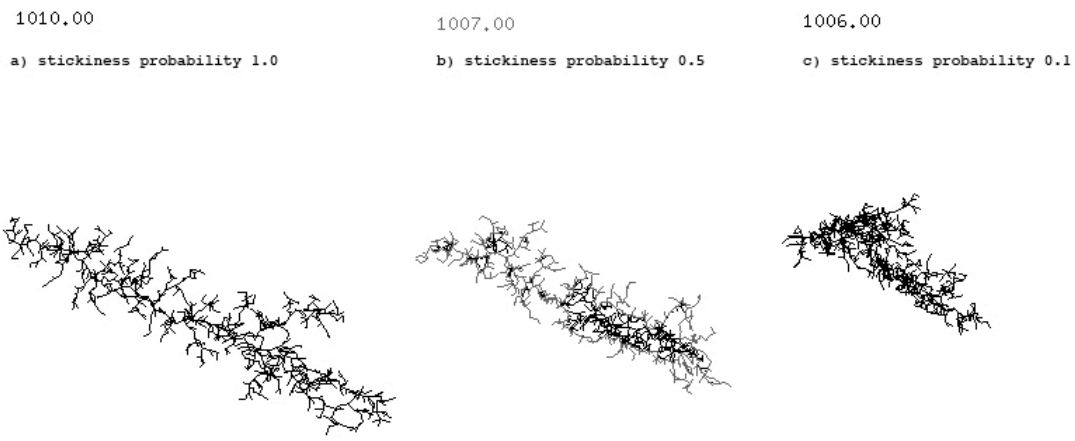


Figure 15: 3D DLA simulation with stickiness probability and angle of diffusing particles set to 30°

After experimenting and understanding the basic principles of the DLA, three-dimensional cylinders replaced the lines. An additional parameter was added to define the radius of the cylinders. Each particle was generated with an initial size, which then increased by a factor each time a new particle was added to the aggregate in order to improve the stability of the system. The result is shown in

figure 16. The addition of this calculation was inevitable as from the very beginning the aim was to develop structures able to be constructed.

Figure 16: 3D DLA simulation with thickness

OpenFrameworks though has restrictions in terms of three-dimensional exportation of objects as it uses OpenGL for the graphic representation. Therefore, soon after the first experimentations on the abilities of the DLA algorithm, the code base was transferred to Java.

3.1.2. Extending DLA

Until this phase, the aggregation was based on randomly diffusing particles. In order to decrease the randomness of the system and generate a structure with more specific properties an alternative method was used. The walker was allowed to search only at specific positions related to the vertices of polyhedra. Below (fig. 17) follows an aggregation of around 1000 particles, in which the direction of the walker is defined according to the vertices of an icosahedron.

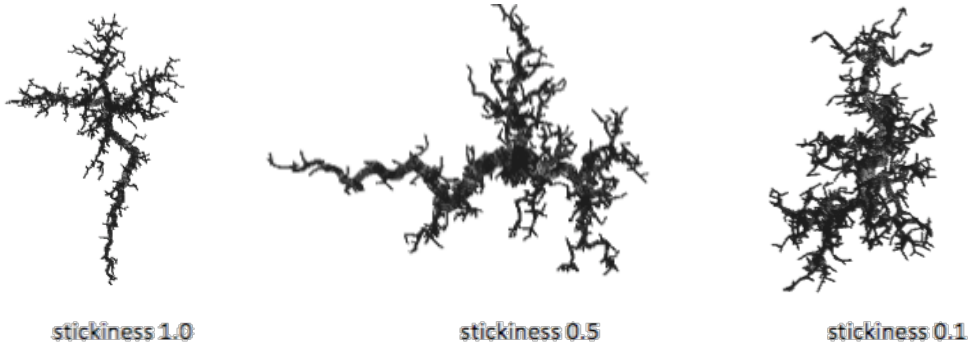


Figure 17: 3D simulation of extended DLA - diffusing particles can take the place of the vertices of polyhedrons

It is obvious, that the aggregation grows in specific directions and particularly to the angles of the applied polyhedron. As a similar approach has not been implemented in the field of the DLA applications, the findings of this phase presented great interest. However, this attempt did not provide a solid answer for developing forms capable for further fabrication in architectural design.

The next step was to connect nearby tubes, to create convex spaces. Likewise above, there is no literature to cover these findings. The resulted lattice exhibits great interest and presents possibilities for further development. Nevertheless, enclosure space is not easy to be achieved because of the nature of DLA algorithm, which only allows for the creation of branch-like structures. The image below (fig. 18) presents an aggregation of 2600 particles.

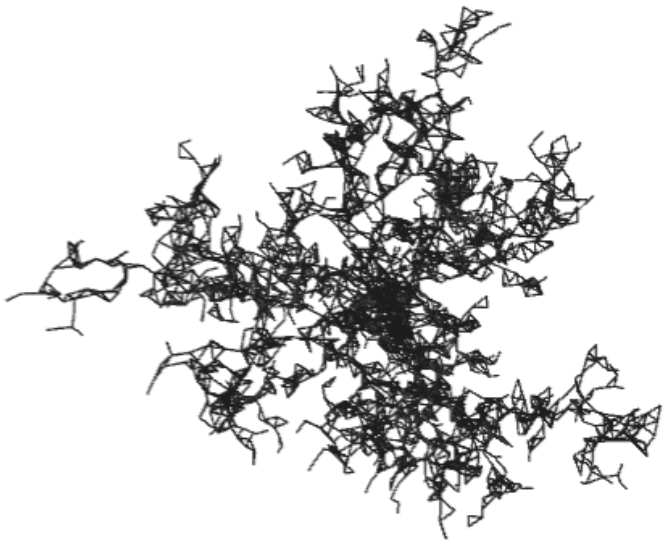


Figure 18: 3D simulation of extended DLA – connection between nearby particles

3.2. Force Analysis

In order to connect the particles in such a way that would allow for measurements about the stability of the structure to be calculated, there was a need for defining some parental relations. During the development of the aggregate each new particle that was added was set as a 'child' to the particle that was attached to. As a result, each branch formed a series of relations that were important for manipulating the structure. Until this phase, the program consisted of three classes: the main one, the Particle class and the Walker class. An important step was to create another Class that would deal with the tubes that connect the Particles. Each tube that was created consisted of two particles: the one generated and its parent. After having this organization, the next step was to calculate the forces that are applied on each tube and on the whole structure in general. The analysis was focused on the weight of the structure and the moment that is produced as a result of it. In order to calculate the weight, each tube had to get a specific identity. For this purpose, a sequential number was applied to every new tube as well as to the generated particles. Concurrently, the centroid and the volume of each tube could be calculated through different functions. The centroid was considered as the place where the total weight was applied and was defined at the middle of the distance between the two particles. The weight of each tube was a vector that started from the centroid and was perpendicular to the XZ plane with a magnitude equal to the mass (volume * density) of the tube. In fact, weight is expressed as the mass multiplied by the g-force but because the acceleration was equal for all the tubes, g-force was set to 1.0 m/s^3. The calculations covered two phases: the first was about the moment applied to a specific joint of the structure from the weight of the whole structure above it; and the second was focused at the transfer of compressive forces within the structure and the moment applied to each joint. For the first procedure, the aim was to calculate the center of mass of the structure and the distance between the center of mass and the joint, which was calculated each time, as the moment is the product of the weight force and its perpendicular distance from the currently calculated joint. To estimate the center of mass (fig. 19, pp. 31) a recursive method was undertaken, which algorithm is described below:

In Class Main:

1. **FOR** each tube
   a. **SET** the calculation mode to false
2. **CREATE** a new Vector CM at (0,0,0), which represents the position of the center of mass of the structure
3. **CREATE** a float TM equal to 0 for the total mass of the structure
4. **CREATE** an integer CP equal to the particle that will be calculated.
   To calculate the moment for the whole structure: **SET** CP=0
5. **CREATE** an Array List of integers TEMP
6. **FOR** each tube (i)
   a. **IF** the particle at the beginning of tube is equal to the particle CP
      THEN add i to TEMP
7. **FOR** times equal to the size of TEMP
   a. **GET** tube(TEMP) and **RUN CALCULATEMASS** function
8. **SET** CM = CM/TM
9. **CREATE** float DIST equal to the distance of CM to CP at the XZ plane.
10. **CREATE** float MOMENT = DIST * TM

In Class Tubes:

CALCULATEMASS function:

1. **CREATE** a vector for the current mass CURM at (0,0,0)
2. **SET** CURM to the center of mass of the tube
3. **MULTIPLY** CURM with the weight of the tube WT
4. **ADD** CURM to CM in the Main class
5. **INCREASE** TM in the Main class by WT
6. **FOR** every child of the tube
   a. **IF** the calculation mode of the child is FALSE
      THEN run **CALCULATEMASS**
7. **SET** calculation mode of the tube to TRUE

To calculate the compressive forces within the structure a similar to the above-described algorithm was applied. The main difference was that the total center of mass was not taken into consideration and the weight vector of each tube was projected to a force along the axis of the tube and its residual one (perpendicular to tube's axis). In this assumption, the force along the axis transfers the compressive force through the structure, while its perpendicular remnant is responsible for the moment at each joint. Figure 20 shows the analysis that was followed for a set of tubes.

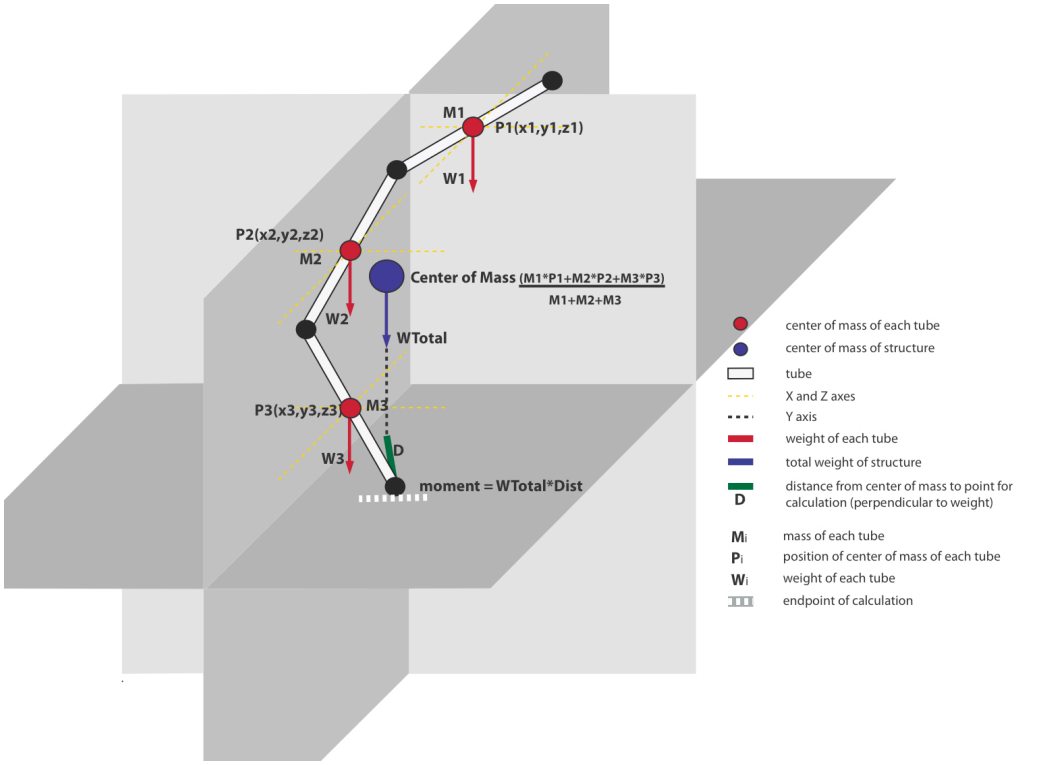


Figure 19: Force Analysis diagram - center of mass

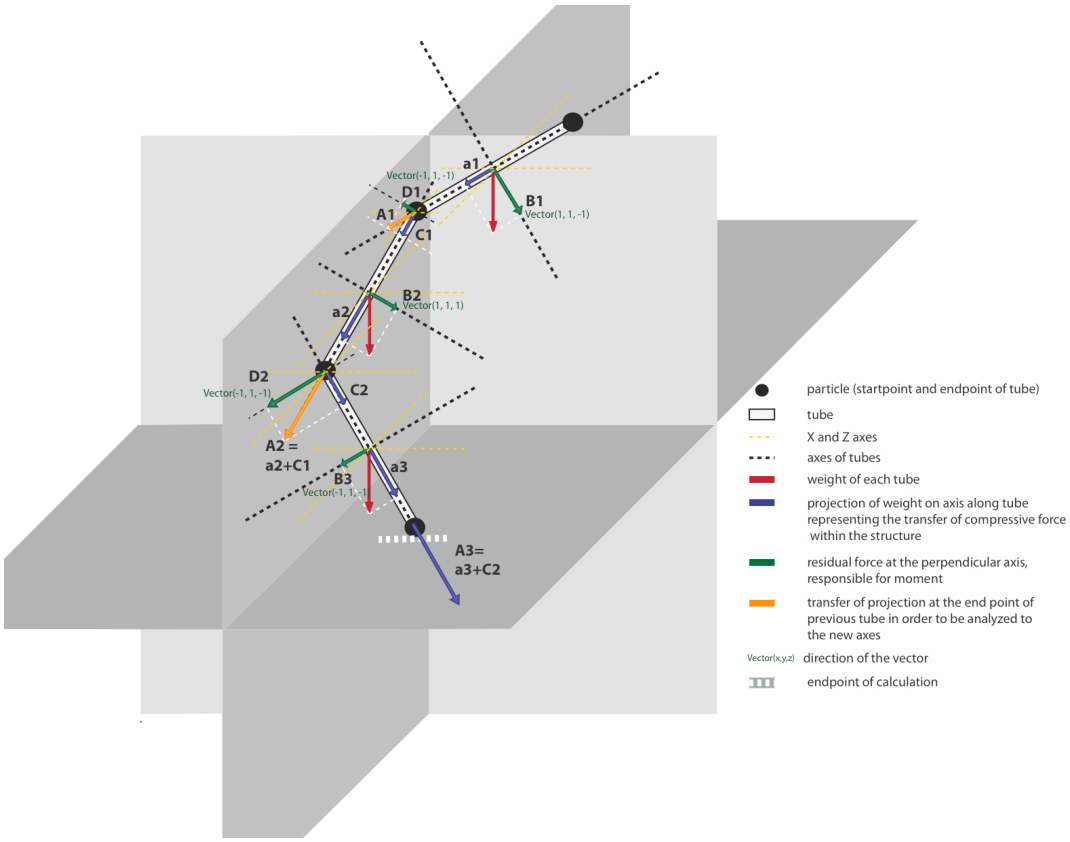


Figure 20: Force Analysis diagram – transfer of forces within the structure

To define the compressive force, a function that calculates the dot product of the weight vector and a normalized vector with a direction from the center of mass of the tube and its starting point was set. As moving from joint to joint apart from the compressive force that was generated from the weight of the current tube, the previous calculated force was again analyzed to the new axes. The perpendicular to the axis of the tube remnants were added at each joint and provided a moment at a specific direction while the compressive forces were added separately. The limitation was set to a specified number and if the compressive force or the moment at a joint was more, the structure above this joint was left to break and fall. The aggregation of the tubes though, did not allow for very stable structures to be done. The results of the above analysis lead to very restricted forms that grew in specific directions. For this reason, the tubes were replaced by polyhedrons attached with aligned faces.

3.3 Diffusion-limited aggregation of polyhedra

Polyhedrons present great ability of managing and transferring the applied forces. A main consideration was to select a polyhedron able to fill space. The main reason for this decision was to decrease the number of the additional constraints in order to maintain the basic principles of DLA algorithm and also to provide an economic module in terms of the space that is created. In contrast to platonic solids (except cube) that require more restrictions in order to attach with aligned faces and do not produce a solid structure, the use of space filling geometry was proved to be more appropriate.

Based on the requirement for a polyhedron that fills space and is capable of efficient force transfer, the truncated octahedron was finally used. A truncated octahedron is an Archimedean, space-filling polyhedron with 24 vertices, 36 edges, and 14 faces. In fact, a truncated octahedron is created by joining two square pyramids together at their bases in order to form an octahedron and then cutting all six corners at one-third of the edge length from each vertex. In a truncated polyhedron, all edges are of the same length and eight of its faces are regular

hexagons while the other six are squares. The ability of the truncated octahedron module to attach at a square face, hexagonal face, or a combination of faces results in a large number of unique configurations if more modules are added to the structural system. Besides, the truncated octahedron is the simplest and the most economical in terms of surface area-to-volume ratio of the space filling systems (Pearce, 1978) and it indicates a high degree of stability because of the triangulations that are formed.

In order to aggregate the truncated octahedron with a DLA method, each particle was considered to transfer a virtual shape with it. The 24 vertices of the truncated octahedron centered at the origin are given below:

(-b, 0, a)	**(0, -a, -b)**	**(-a, -b, 0)**	**(b, 0, a)**	**(0, -b, a)**	**(a, 0, b)**
(-b, 0, -a)	**(0, -a, b)**	**(-a, b, 0)**	**(b, 0, -a)**	**(0, -b, -a)**	**(a, 0, -b)**
(-b, a, 0)	**(0, a, b)**	**(-a, 0, b)**	**(b, a, 0)**	**(0, b, a)**	**(a, b, 0)**
(-b, -a, 0)	**(0, a, -b)**	**(-a, 0, -b)**	**(b, -a, 0)**	**(0, b, -a)**	**(a, -b, 0)**

where **a** is the edge length of the truncated octahedron multiplied by the square root of 2, which actually is the distance of the centroid to the square face, and **b** equals to 0.5 * **a**.

In order to aggregate the polyhedron with aligned surfaces a set of additional constraints were set to the algorithm. The walker was still allowed to search randomly in a sphere around the seed particle. If it was found to be at an appropriate distance and also if it had the necessary space to put a polyhedron without overlapping with another polyhedron earlier attached, then it attached a new polyhedron to the aggregate at the current position. Otherwise, it was still searching until finding a place to attach a truncated octahedron. The distance was defined in two different conditions. To attach the polyhedrons with aligned square faces the distance should be two times the distance of the centroid of the polyhedron to the center of its square face (square radius) while for acquiring adjacent hexagon faces the distance was set to two times the distance of the centroid to the center of the hexagonal faces (hexagonal radius). The figure below (fig. 21) shows the two different cases.

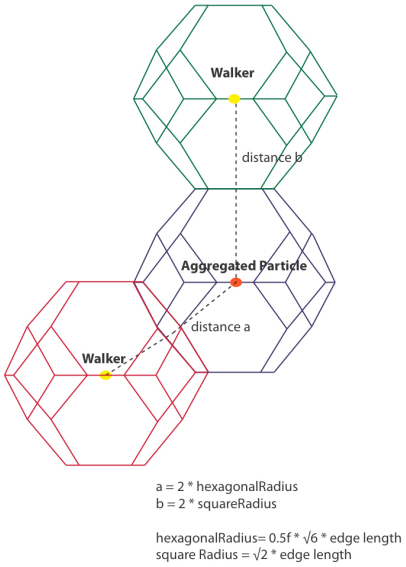


Figure 21: adjusted truncated octahedrons with aligned faces

3.3.1 Torque calculation

Once the constraints of the aggregation were utilized, the next step was about the calculation of the forces within the system. As discussed above, the triangulations that are formed within a space filled with truncated octahedrons allows a high degree of stability and an equal transfer of the compressive forces. For this reason, rather than calculating the compressive forces another approach was followed. Each shape consisted of a 36 tubes that connected the vertices of the truncated polyhedrons. The tubes of each shape were at first created with a specific size, which was increased by a specific factor each time another truncated octahedron was attached to the shape or to its 'children'. By increasing the size (diameter) of the tubes, the resistance of the structure to compressive forces was stabilized.

Regarding the calculation of the moment, the initial idea was similar to the method that was followed for the tubes but the calculation affected the evolution of the whole structure. At first, the center of mass of each truncated octahedron was set at the centroid of each shape. A recursive method was then followed to calculate the center of mass as well as the mass of the structure above every particle. When the walker attached a new shape to the aggregate, a calculation of the moment was

done. If the moment was more than a specified value then it removed the shape. Moreover, a factor for increasing the threshold while moving from the exterior branches to the interior and the base of the structure was added. This addition was necessary as the 'older' branches were heavier and thus presented greater resistance to the applied forces (fig. 22, pp. 36). The algorithm that was applied for the calculation of the moment is described below:

In Class Main:

1. **CREATE** a Boolean calcCM
2. **CREATE** an integer S equal to sequential number of the last created shape
3. **CREATE** an integer T equal to 0
4. **CREATE** a float MOMENT
5. **CREATE** a float THRESHOLD
6. **CREATE** a float MOM_STEP
7. **WHILE** TM is greater than 0
   a. **FOR** each shape
      i. **SET** the calculation mode to FALSE
   b. **SET** the center of mass CM at (0,0,0)
      c. **SET** the total mass of the structure TM equal to 0
      d. **FOR** times J equal to the number of shapes
         i. **IF** S is equal to the sequential number of the shape
            THEN 1. **SET** T=J
                     2. **BREAK**
      e. **IF** the shape has more than one child
         THEN multiply THRESHOLD by a constant factor
      f. **RUN CALCULATEMASS** function on SHAPE(T)
      g. **SET** CM = CM/TM
      h. **CREATE** float DIST equal to the distance of CM to CP at the XZ plane.
      i. **CREATE** float MOMENT = DIST * TM
      j. **IF** MOMENT> THRESHOLD
         THEN TM = -1*TM
      k. **IF** TM < 0 **THEN** BREAK
      l. **ELSE** i. **IF** SHAPE(T) has a parent
            THEN SET S equal to sequential number of the parent of SHAPE(T)
         ii. **ELSE BREAK**
8. **IF** MOMENT – 'previous MOMENT' > MOM_STEP
   THEN SET SHAPE(J) STYLE = lightweight
9. **IF** TM < 0 **THEN SET** calcCM = FALSE
10. **ELSE** a. **SET** calcCM = TRUE
      b. **SET** 'previous MOMENT' = MOMENT

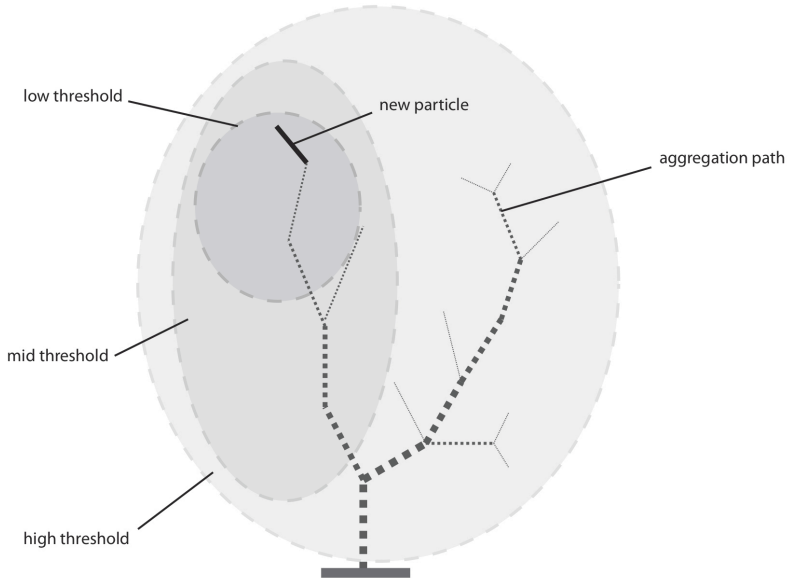


Figure 22: Path of the aggregation – diagram of increasing torque thresholds

After a number of aggregated polyhedra, the structure reached equilibrium. The number of the aggregated polyhedrons was dependant on the maximum moment that the structure could stand as well as to the stickiness probability. The denser the structure was, the most stable it proved to be.

At the latest phase of the algorithm an improvement was made in order to manage its stability more efficiently. During its evolution, there were two states of placing shapes. The first was about placing solid truncated octahedrons while the second referred to wire-framed truncated octahedrons structured with tubes. The selection was based on the stability of the structure. If the difference of the moment of the structure was much greater than the previous, a wire-framed polyhedron was placed. The wire-framed polyhedrons have less volume and thus apply less force to the structure. Otherwise, a solid polyhedron was added to the aggregate. This attempt increased the number of the aggregated polyhedra in order to reach equilibrium and also provided an alternative of interior and exterior spaces.

Having a structure that is self-sustained was of crucial meaning for this research as it allows the placement of the structure onto the existing infrastructure without the need of additional structural support.

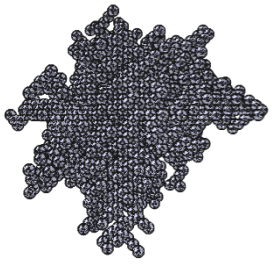
4. Testing and Results

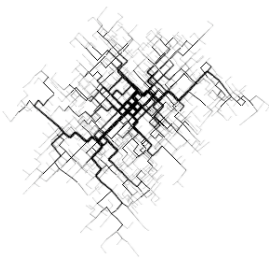
4.1. Case study 1 - Overview of fractal dimensions

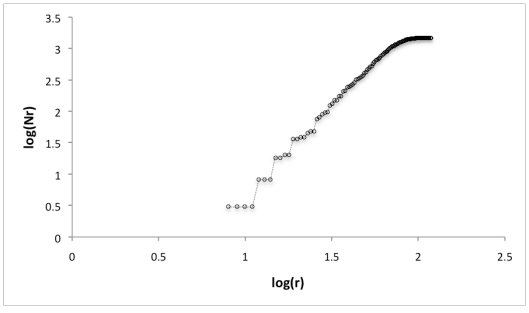
This section presents a series of experiments about the fractal dimension of the structure. The results show how the fractal dimension of the structure is affected by different parameters within the algorithm. Each of the below aggregate grew in open space and without any structural limitations.

4.1.1. First Experiment - Changing the direction of the walker

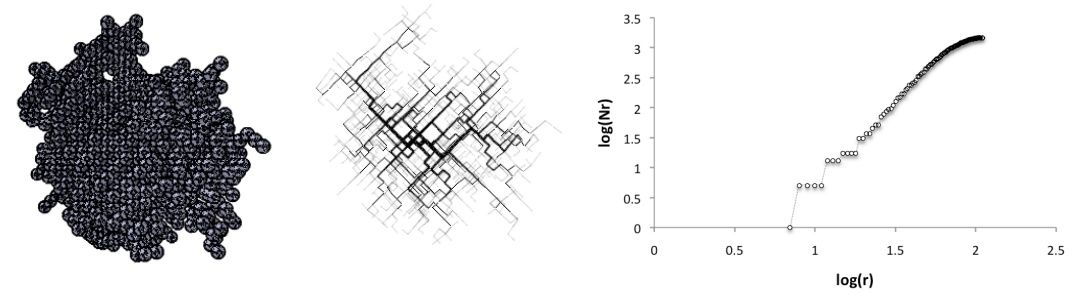
In this first experiment the parameter that changes in all cases is the angle θ, in which the walker is allowed to search for aggregate particles. The aggregation consists of 1450 particles and the stickiness probability is equal to 1. Images at the first column show the aggregation model, at second column the path of the aggregation whereas third column is the graph representation of the fractal dimension. As discussed before (see section 2.3.), the fractal dimension can be calculated through the equation $N(L) = L^D$, which can be translated to $\log(N(L)) = D*\log(L)$. After plotting the extrapolated data of N(L) and L, the slope of the line gave the fractal dimension for each case.



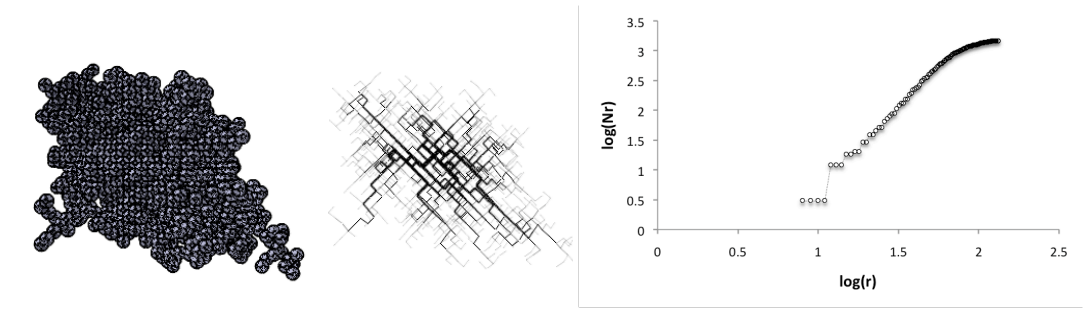




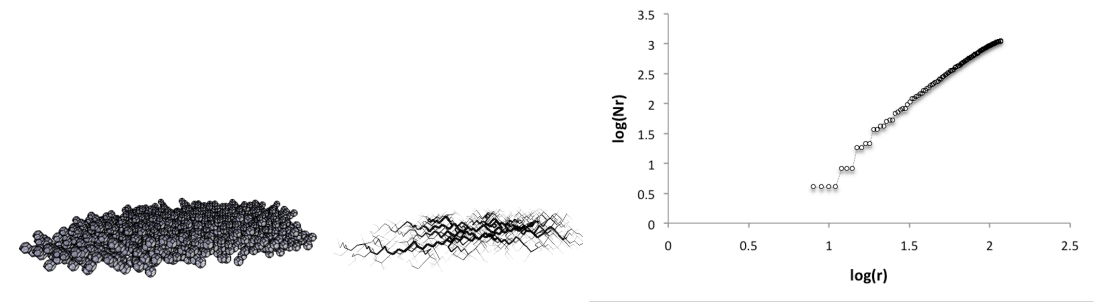
θ = random from 0° to 360° **D = 2.475**



θ = 60° **D = 2.469**



θ = 30° **D = 2.27**



θ = 180° **D = 2.166**

Figure 23: 3D DLA with truncated octahedrons and diagrams - the relation between the fractal dimension and the angle of diffusing particles

The results show that while the angle between X, Y, Z axes decreases, the fractal dimension decreases too. This result can be used for filling spaces with or without changing their fractal dimensionality and character. For instance, in a specific given infrastructure the selection of the appropriate growth procedure can help for extending the built environment by retaining the character of its landscape.

4.1.2. Second Experiment - Changing the stickiness probability

The second phase of the experimentation on fractal dimensions examines the effects of changing the stickiness probability. The aggregates, which consist of 1500 particles, grew in open space, without any structural limitations and the angle θ of the walker was set randomly from 0° to 360°.

stickiness probability : 1.0 **stickiness probability : 0.1**

Figure 24: 3D DLA models of truncated octahedrons with stickiness probability

In three-dimensional typical DLA models with stickiness probability, the fractal dimension can takes values equal to 2.50 ± 0.15 (Vicsek, 1992). The diagram below shows the relation between the stickiness probability and the fractal dimension of the structure proving that the aggregation of truncated octahedrons maintains the same character with typical DLA models.

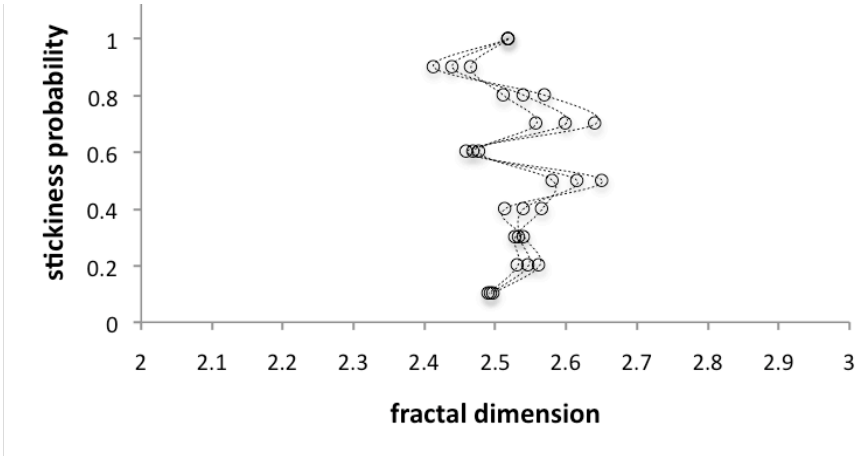


Figure 25: diagram showing the relation of fractal dimension and stickiness probability for 3D DLA with truncated octahedrons

4.1.3. Third Experiment - Alternative algorithms

As discussed at section 2.3 Diffusion Limited Aggregation has several extensions. Within this research apart from DLA, diffusion-limited deposition, ballistic aggregation and ballistic deposition were also studied.

- Diffusion-limited deposition (DLD)

In order to replicate diffusion-limited deposition instead of a single seed, a lattice was set as the initial seed of the structure. Because the relations in the layers of the structure are the same as in the DLA model (Vicsek, 1992) as they are both based on the same algorithmic procedure, the fractal dimension of the models resulted to be equal to this of DLA models.

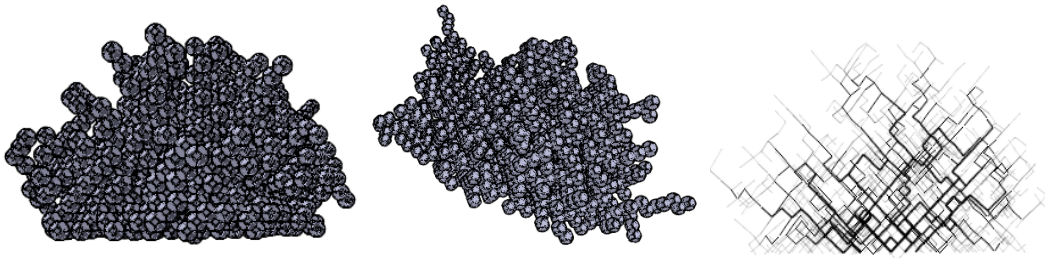


Figure 26: 3D DLD models with truncated octahedrons

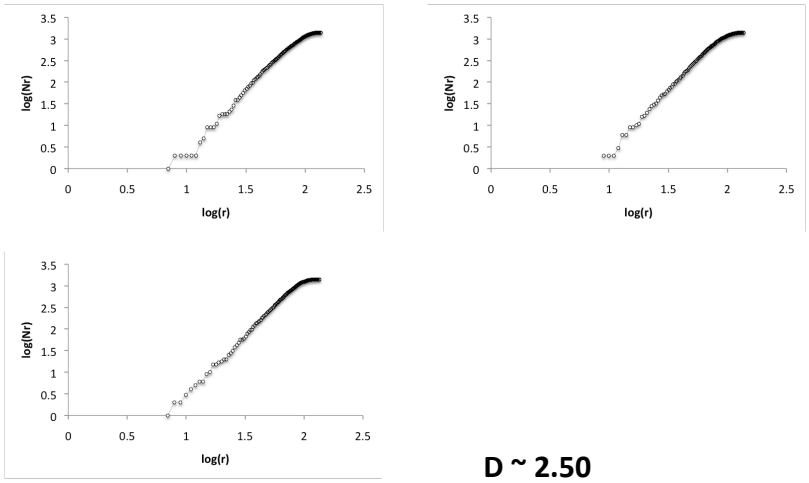


Figure 27: diagrams showing the fractal dimension of 3D DLD models with truncated octahedrons

- Ballistic aggregation (BA)

For ballistic aggregation and ballistic deposition (see below) a similar algorithm was used but instead of giving a random direction to a walker, a group of particles (walkers) is moving along straight trajectories until they encounter the growing aggregate and stick to its surface. This lead to specifically orientated structures that would be very useful in very dense environments with a lot of constraints, where the empty space is provided only to specific directions.

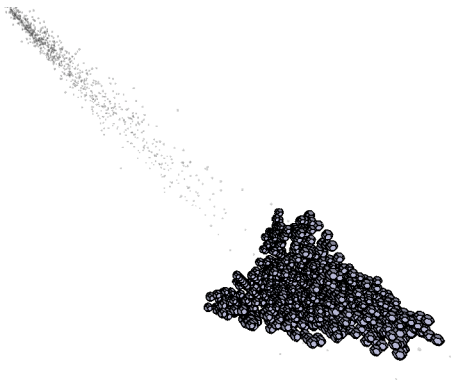


Figure 28: simulation of ballistic aggregation model in Java

As discussed in section 2.3 ballistic aggregation is implemented on a single seed. The resulted forms exhibited great interest and complexity especially in the way that the truncated octahedrons were attached to each other at specific directions. However, the presence of large, extended holes of various sizes negates the characteristic of self-similarity. Consequently, ballistic aggregates are not fractals (Vicsek, 1992; Liang and Kadanoff, 1985).

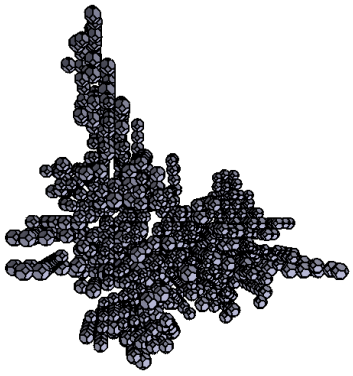


Figure 29: simulation in ballistic aggregation model in Java

- Ballistic deposition (BD)

In ballistic deposition the particles were allowed to move along parallel straight lines until they contact either a particle in the deposit or reach the initial surface. The resulted models consist of columnar structures connected to the same root at the surface but the structures as a whole proved to be rather bulky to support parasitic character.

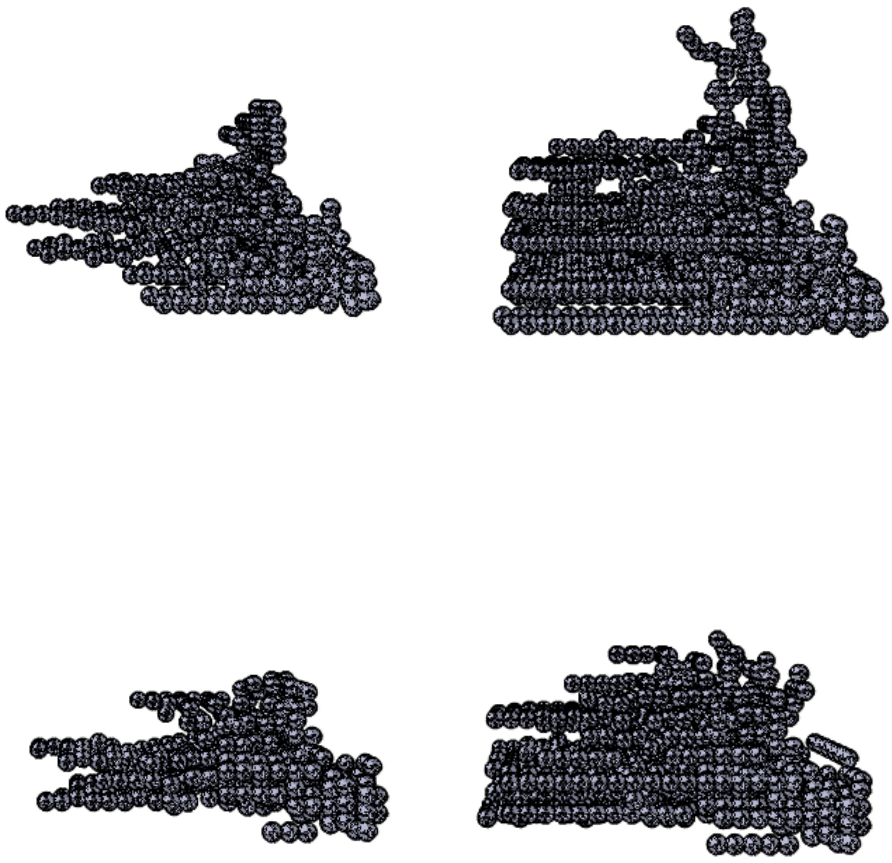


Figure 30: simulations of ballistic deposition models in Java

4.2. Case study 2: Structural limitations

4.2.1. First Experiment - size of the aggregate

This experiment studies the number of aggregated particles for DLA, DLD, BA and BD models. The simulations had the same parameters and moment thresholds. After a number of simulations the extrapolated data showed that the structures of the diffusion-limited aggregate are more stable than ballistic aggregates.

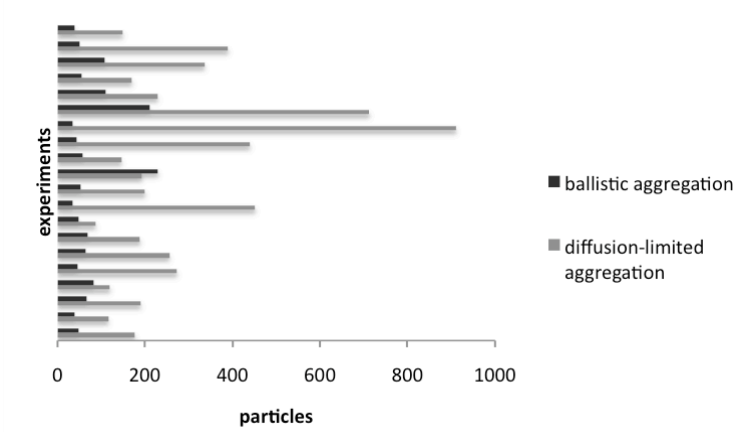


Figure 31: diagram showing the difference of the size of the aggregate between BA and DLA models after applying structural restrictions

Diagrams below show the plotted data and express the relation between the maximum moment that the structure could afford and the number of particles that have been aggregated.

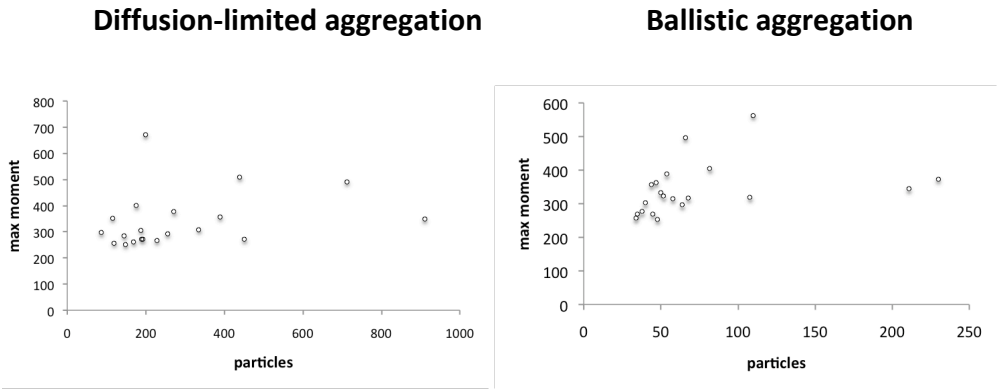


Figure 32: diagram showing the relation of the aggregation size and torque for DLA and BA models

In comparing diffusion-limited deposition and ballistic deposition, the results were similar i.e. DLD models exhibited greater stability than ballistic deposition models.

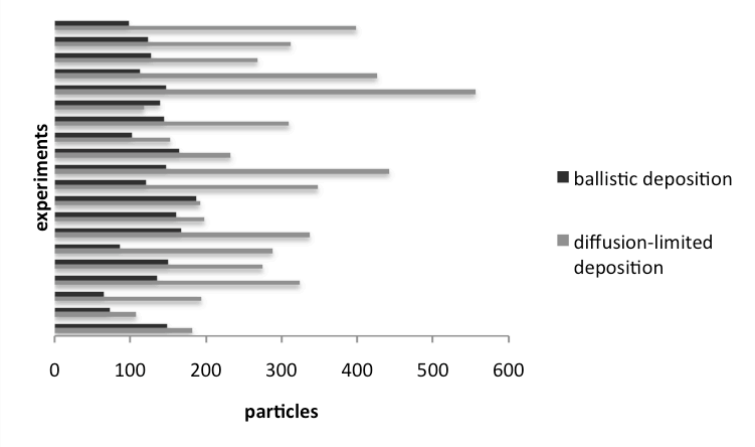


Figure 33: diagram showing the difference of the size of the aggregate between BD and DLD models after applying structural restrictions

4.2.2. Second Experiment - relation between mass and moment

Examining the relation between mass and moment can strengthen the above result. As shown in the diagrams below, the increase of the mass in ballistic aggregation causes rapid increase of the moment as the mass is distributed to a specific direction. In contrast, the mass in diffusion-limited aggregates is delivered more efficiently and thus the generated moment increases gradually.

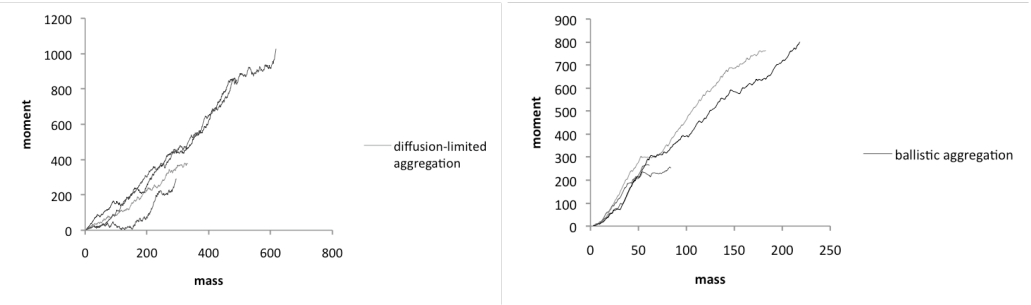


Figure 34: diagrams showing the relation of mass and torque for DLA and BA models during the growth process

4.2.3. Third Experiment - Increasing stability

As discussed in the methodology, in order to increase the size of the aggregate and also create an opportunity for both interior and exterior spaces, a wire-framed truncated octahedron and a solid one were placed. The wire-framed truncated octahedrons maintain the structural stability as it triangulates space likewise the solid modules. However, their weight is decreased by a considerable amount. The selection was dependent on the increase of the moment. If the addition of a solid shape tended to increase the moment to a great extent compared to previous calculations, a wire-framed shape would be placed at the aggregate instead. The diagram below shows the effect of this implementation on the relation between mass and moment during the growth process of three DLA models with different structural limitations.

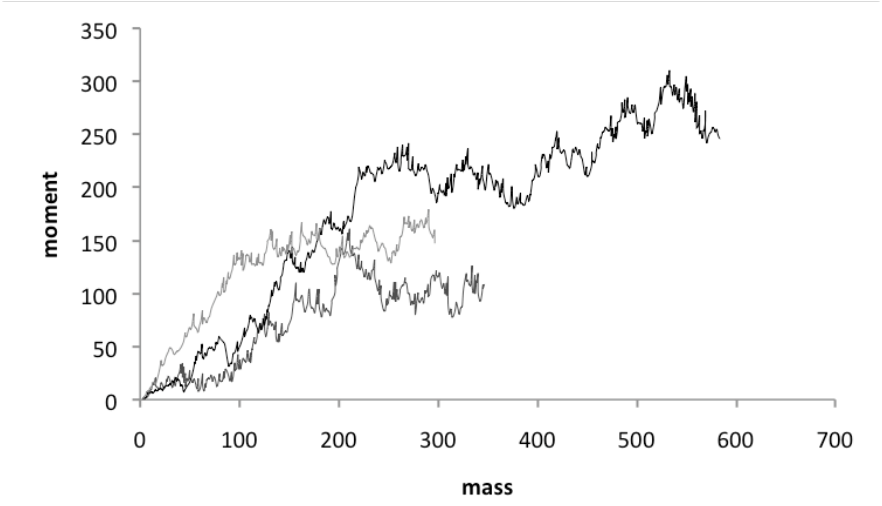


Figure 35: diagram showing the relation of mass and torque of a DLA model during the growth process after combining closed and open shapes

It is obvious that the placement of the wire-framed truncated octahedrons decreases the moment during the evolution process and slightly stabilizes the structure. This implementation can provide larger aggregates within strict structural limitations that the use of different materials may require.

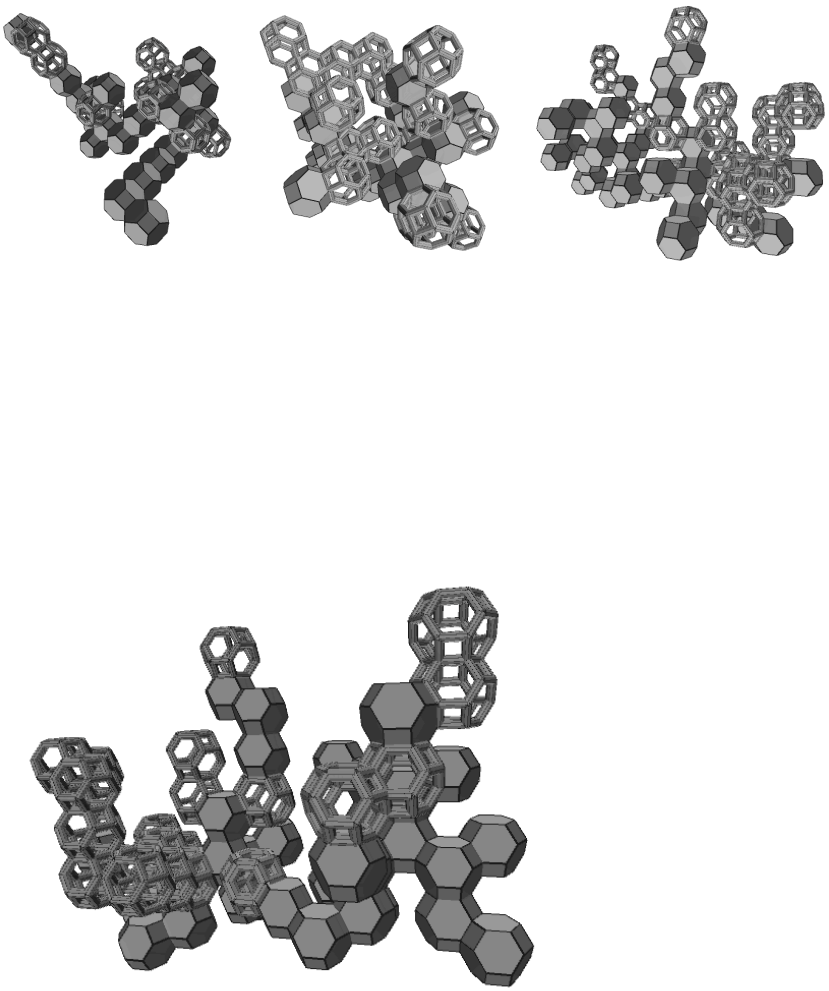


Figure 36: 3D simulations of a DLA model with closed and open spaces after applying structural restrictions in Java.

5. Adaptation

This chapter is about implementing the previous findings along with the methodology described in order to examine how these models can adapt to different contexts.

In order to examine how the aggregated parasite will adapt to its context, a new Class was created, which organizes the generation of the three-dimensional geometry that was placed as a simulation of the built environment. Additionally, a simple algorithm for collision detection was implemented to prevent particles from inserting into the surrounded boxes. As resulted from the above experiments, diffusion-limited aggregation provides better stability, which furthermore increases with the simultaneous use of wired-framed and closed space. It has to be clear that the implementations could follow a directional evolutionary process by applying variations to the angle of the walker and also vary in terms of density as shown in the experiments at section 4.1.1 and 4.1.2. However, for this particular experimentation, the implementation is based on a typical solution of a stickiness probability equal to 1.0 and an angle randomly set from 0° to 360° degrees. The edge length of each truncated octahedron was set to 1.5m in order to provide a reasonable space of around 38 $m^{3.}$

Below follows a series of case studies that present the adaptation of the parasitic structure within different infrastructures and examine the changes of the fractal dimension of the context as well as the resulted form.

5.1. First Experiment - DLA model grown in open space

In this first experiment, the structure was allowed to grow in an open space environment. The fractal dimension of the existing infrastructure was defined through calculating the box-counting dimension of the object (Addison, 1997). The slope of the line of the plotted data was equal to 2.637. The aggregate reached equilibrium after 72 particles, of which 40% were exterior spaces, and had a fractal

dimension of 2.615. The calculations of the fractal dimension after the placement of the parasite exhibited a 0.4% increase providing a functional interior space of 1680m^3 and an exterior space of 1069 m^3.

As shown in the images below, the resulted form supported the existence of free space at the ground level and also the free spaces within the structure itself allows the creation of well-ventilated environments.

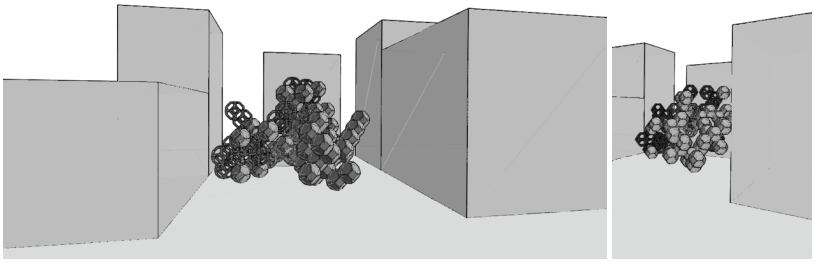


Figure 37: simulation of the parasite grown in open space in Java

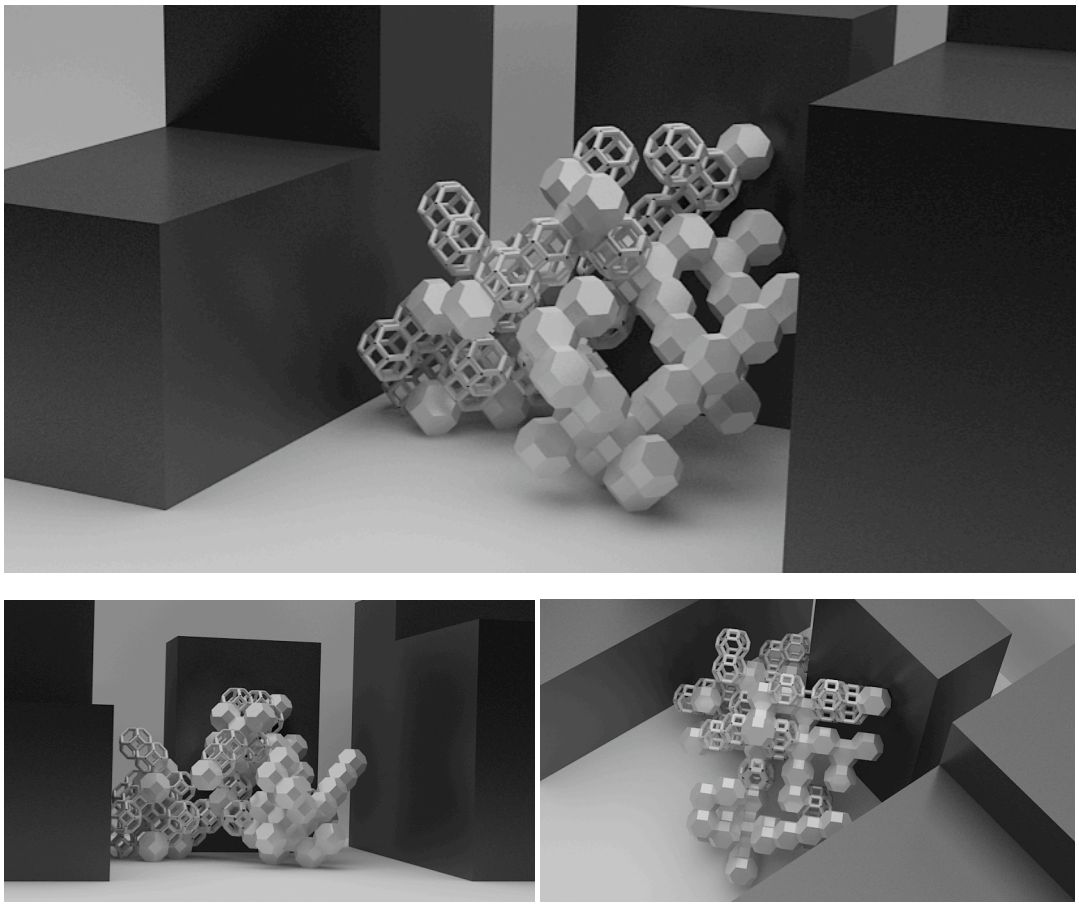


Figure 38: 3D models of the parasite exported and rendered

5.2. Second Experiment - DLA model grown in semi-restrictive area

In the second phase the fractal dimension of the environment was increased to a value of 2.86. Additionally, the distance in between the buildings was decreased by 50% and the one box was elevated to examine the adaptation in a rather restrictive area regarding the addition of supplementary space. The structure reached equilibrium at a size of 99 particles, of which 45% were related to exterior spaces of a total 1718m^3 and 2062m^3 of total interior space. The fractal dimension of the structure was 2.6 increasing the overall dimension 0.79%. Likewise in the case above, the resulted form benefits from the available space without transforming incredibly the functionality of the initial space. The holes at the ground level support a passage between the buildings and also the empty space below the elevated box has efficiently been filled allowing significant movement and ventilation.

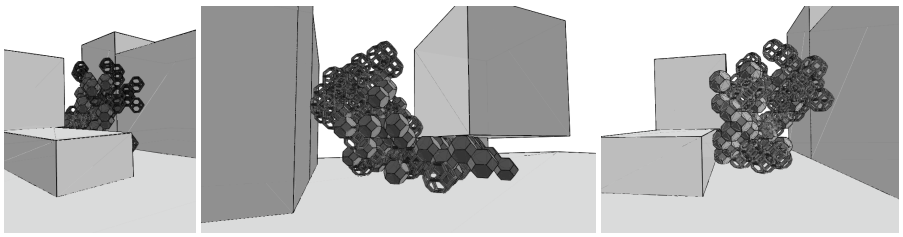


Figure 39:simulation of the parasite grown in semi-restrictive area

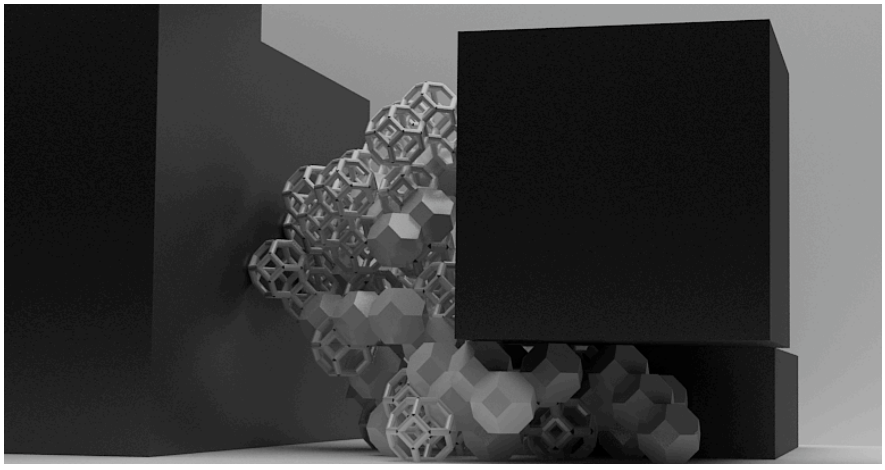


Figure 40: 3D model of the parasite exported and rendered

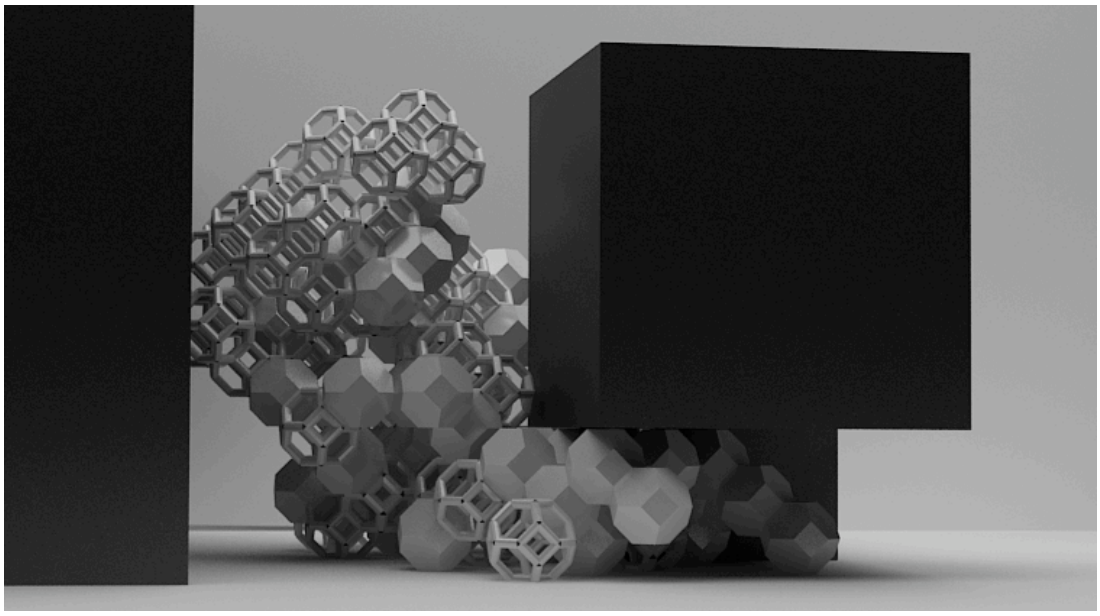


Figure 41: 3D model of the parasite exported and rendered

5.3 Third Experiment - DLA model in highly restrictive area

This third case examines the adaptation of the structure at a very strict area. The dimension between the buildings was decreased 40% compared to the second experiment. Also additional boxes have been placed as an attempt for simulating empty spaces that are actually out of use and usually created at the backyards of very dense structured building blocks. The fractal dimension of the calculated environment was 2.76 and followed by an increase of 0.05% after the development of the aggregation model. The aggregate reached equilibrium after 28 particles, 25% of which were wire-framed providing 267m^3 of exterior space and 802m^3 of interior.

The capability of the structure to add such an amount of space within so restrictive constraints while maintaining ventilation and passages to the ground level is more than valuable for dealing with spaces of similar character.

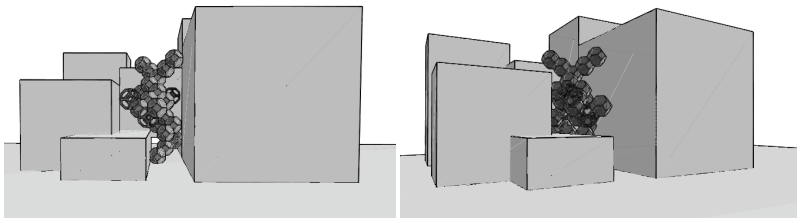


Figure 42: simulation of the parasite grown in highly restrictive area

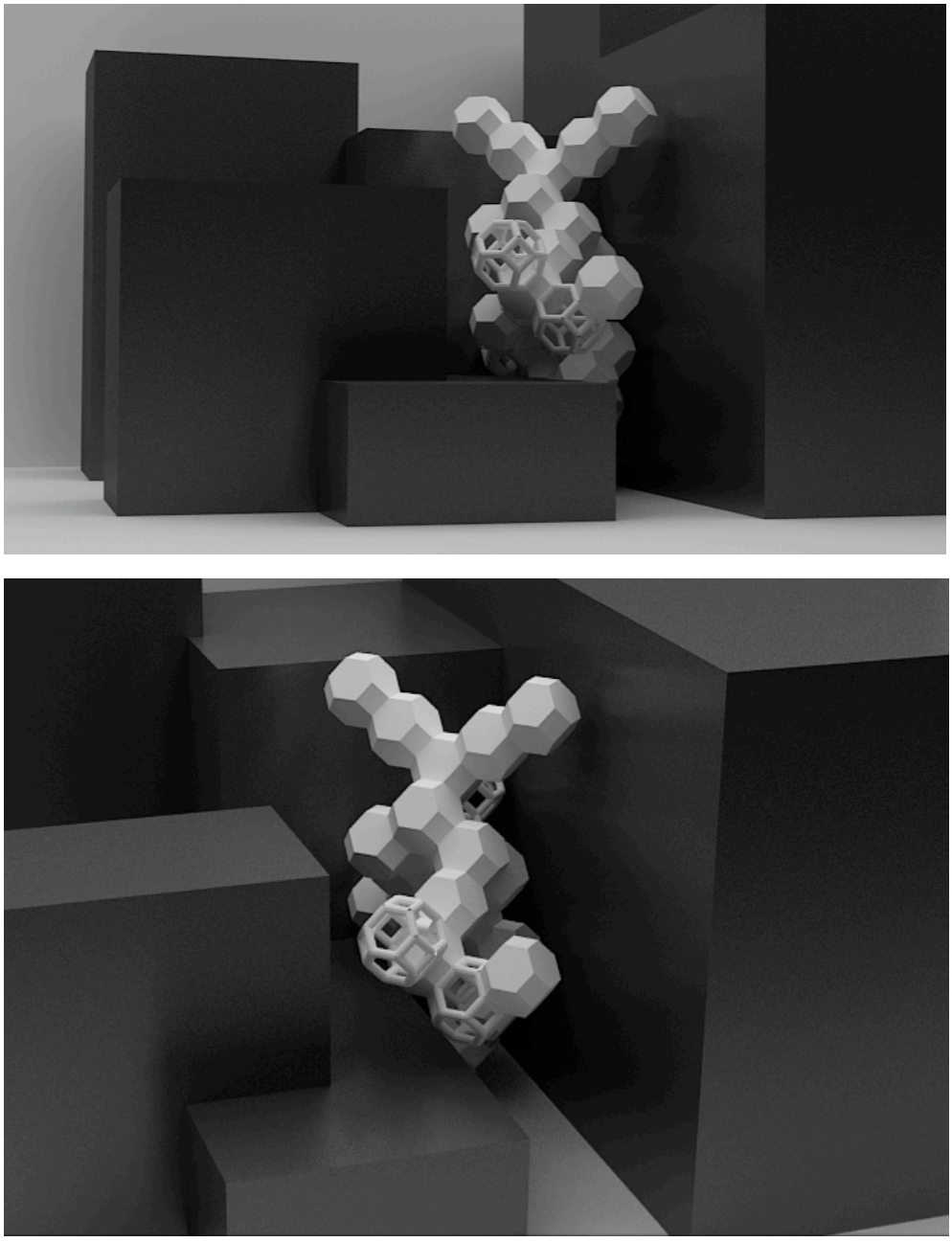
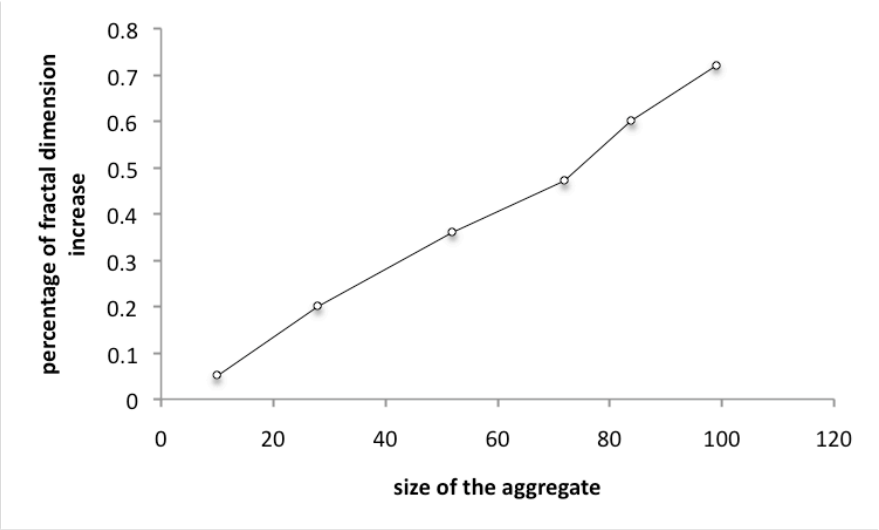


Figure 43: 3D models of the parasite exported and rendered

5.4. First Results

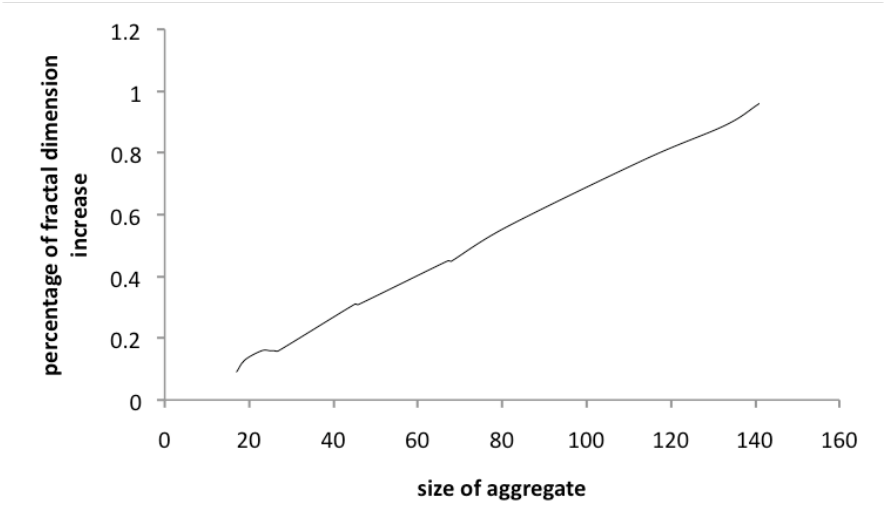


The above results as well as the results of other three similar experiments provided an average 0.0073% increase on the fractal dimension of the existing building environment, dependent on the specific context and thus on the size of the aggregate. Because, the existence of such different parameters does not provide a unique result to be obtained, a final experiment was set based only on the examination of the fractal dimension's increase.

5.5 Fourth Experiment - DLA models grown in same environment with different thresholds.

This final attempt was focused on the development of aggregation models in the same context that was set for the third experiment but with different thresholds regarding their resistance in torque. As a result, the size of the aggregate was varying. The aim was to examine in what degree the fractal dimension of the existing infrastructure will increase after the placement of the parasite. Being aware of the effects of the parasite to its context was necessary for ensuring that its placement will not influence in a negative way the spatial qualities of the existing space. The extrapolated data showed that for an environment with a fractal

dimension equal to 2.76, the increase is equal to an average value of ~0.0066 % for every added truncated octahedron with edge length equal to 1.5m and volume = 27√2 m^3. The diagram below exhibits the outcome of the plotted data.



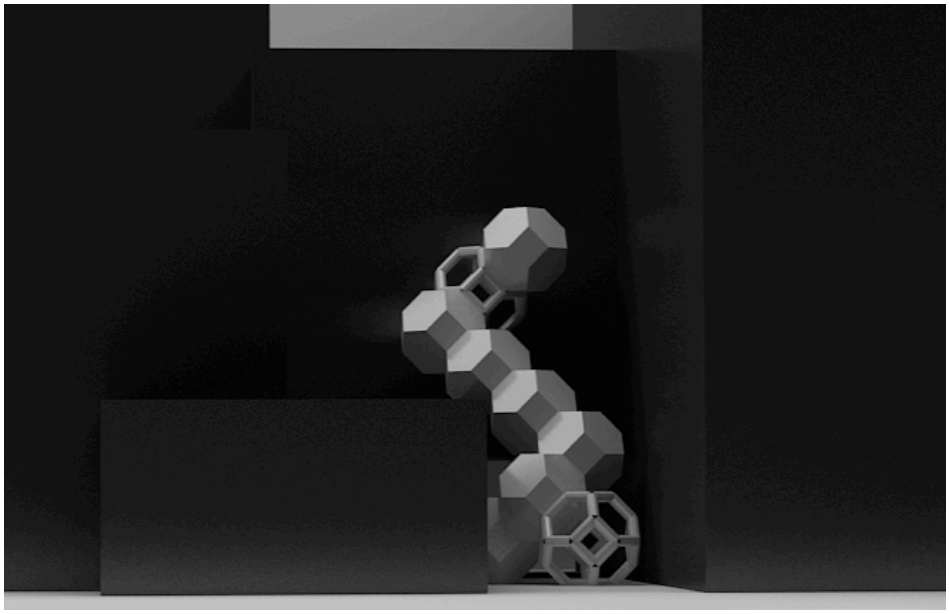


Figure 44: 3D model of the parasite with low torque threshold exported and rendered

6. Discussion and Evaluation

6.1. Overview of findings

According to the above results, the aggregation of truncated octahedrons maintains the characteristics of a typical DLA model in three-dimensions. Moreover, it was showed that through slight changes in the algorithm lot of alternative forms could result. For instance, the modification of the angle in which the diffusing particles are moving or changes to the stickiness probability can produce different forms. The selection of these parameters has to be decided in respect to the existing infrastructure in order to obtain a better adaptation.

Moreover, extending diffusion-limited aggregation to include a structural analysis regarding the torque, which is created from the applied weight to the branches, produces a self-organized structure that reaches equilibrium in response to its context. The analysis on the stability of the structure proved that diffusion-limited method produces more stable structures than the ballistic method. Examining the relation of mass and moment during the evolutionary process further strengthened this finding. The combination of interior and exterior spaces demonstrated an even better capacity in terms of stability and also provided a capability of implementing these structures into architectural design as unique solutions for creating space. The ability of the structure to be self-sustained offers the possibility of extending space without influencing the stability of the existing infrastructure.

Finally, the adaptation of the structure within its environment exhibited great interest. For a moderate aggregation, the parasite slightly increased the overall fractal dimension. Besides, the branching structure of the resulted forms ensured well-ventilation and supported movements at the ground level. Because of the geometry of the aggregation the extension of space was applicable even to very restrictive areas, where traditional methods of architectural design would demand for overpriced solutions. Overall, these fractal structures provide the potential for extending space on the one hand, without increasing that much the fractal

dimension of their context, and on the other hand, at locations were additions are not easy to be implemented.

6.2. Critical assessment

The method presented in this study is related to the research question that was addressed. The aim was to generate a self-sustained parasitic structure able to adapt to its context by creating aggregation forms. The DLA algorithm proved to be satisfactory for this particular research. On the one hand, as DLA has not been applied in architectural design, the results of this research showed that it can be a basis for the creation of emergent and self-organizing forms providing new possibilities of architectural spaces to be further examined. On the other hand, the branching forms that were developed exhibited great interest on how an architect can fill space without changing the functionality of the existing infrastructure. The holes that emerged at the deposits of the structure allowed for well ventilation and movements to be achieved. Moreover, the implementation of a basic structural analysis as an extension within the DLA algorithm offered the development of self-sustained structures as intended to. This result can support the idea of flexible and temporary accommodations that parasitic architecture examines (Allen et al., 2003). Additionally, the production of modular systems can provide low-cost solutions for dealing with the problems of structural density and nomadic lifestyle.

The hypothesis of this research was that through the use of the DLA method, the functional space within cities would extend without transforming remarkably the character of its landscape. Additionally, the inefficient open space in the backyards of building blocks would be rejuvenated and attain a practical usage. It can be pointed out according to the results of the experiments that the hypothesis has been partially fulfilled. Due to time limitations, the experimentation of the effects of the parasite on its environment landscape was limited. It is essential for a large amount of experiments to be set in order to attain a unique result, as the field of DLA implementations in architectural design has not been examined. Moreover, the analysis on the structural behavior of the structure was based on the tendency of

the weight to actually break a branch or dislocate the object. But, before fabrication a finite element approach would be vital. A complete structural analysis, which would include for example, analysis on the strength of materials would likely offer the emergence of different forms. In fact, the materiality of the structure was not defined, as the intent was not to produce a specific solution but a strategy for exploring space extensions. Finally, there is uncertainty regarding the capability of DLA algorithm to respond to further extensions. Specifically, in order to produce a fully adaptive system able to respond to current needs a lot of additional constraints should be implemented within the algorithm. For instance, placing the parasite in the internal of building blocks to extend the already existing space for residents' usage means that the openings of the existing buildings should be matched with the openings of the DLA structure to ensure passage. Such constraints might limit the potential of the DLA to follow a stochastic process in order to evolve and adapt as a self-organized structure and might cause a rather deterministic solution.

6.3. Further work

Based on the limitations listed above, further work will basically focus on examining these possibilities. As an immediate intent, material specifications will be implemented followed by a finite element approach in response to diffusion-limited aggregation evolutionary process. A complete structural analysis will provide forms capable to be assembled. Moreover, a specific design for the jointed parts is inevitable. Examining the way that the edges of the truncated octahedron will fit to one another offers the possibility of easy construction. Besides, it would fulfill the second hypothesis, which assumes that the structure would be capable to re-organize its form in respect to current needs. In this case, residents would participate in the growth process by adding or removing parts of the aggregate giving a breathing character to the city in which they live. After defining the constructive mechanisms and ensuring the ability of the structure to be assembled, the next phase will include more experimentation on the effects of the parasites to

the fabric of cities. Further experimentation will ensure that the additional space will not affect negatively the functionality and character of the existing space. Furthermore, it will create a framework able to be applied in different locations with awareness of the modifications that will emerge.

Moreover, it is very important to examine whether additional constraints will affect the behaviour of the DLA algorithm. If so, it is important to explore ways for ensuring better adaptation in respect to the evolutionary process of diffusion-limited aggregation. Finally, apart from the truncated octahedron, other geometries can be explored so that new architectural possibilities would emerge. The examination of spaces with alternative geometry might provide solutions of more economic and functional constructions.

7. Conclusion

This thesis has addressed the development of self-sustained parasitic structures that evolve by creating aggregation forms well adapted to their context. The innovation of the research involved on the one hand, the biological paradigm of fungal colonies, which adapt to the changing environmental conditions providing a meaningful example of ecological relations such as parasitism, in nature. On the other hand, the concept of flexible and temporary structures as has been applied so far in implementations of parasitic architecture were a starting point for considering the necessity of solutions that ensure the extension of space in dense structured cities.

The methodology followed was based on the DLA algorithm extended to include a structural analysis and also to allow the aggregation of space filling geometry. The project started from the exploration on the abilities of tubular structures; continued with the insertion of truncated octahedrons and a structural analysis of the system in order to maintain structural integrity; and resulted to an examination of its ability for adaptation into physical environment as the main objectives initially required.

The hypothesis of the research was based on the ability of DLA implementations to produce space as extension to existing infrastructure without remarkable modifications of its character and also to develop self-organized structures that do not need structural support. By conducting a series of experiments, the capacity of the algorithm was proved successful in developing self-sustained structures able to extend space in locations with spatial limitations. During the structural analysis, extensions of the DLA (diffusion-limited deposition, ballistic aggregation and ballistic deposition) were also studied and compared as an attempt for finding the best solution. The estimation of the results showed that diffusion-limited method is more capable for providing stable structures. Further experimentations proved that the additional space advantageously offers new accommodations with neutral modifications to the functionality and character of the existing infrastructure. However, this outcome has a partially local character. In order to apply such a

strategy for space extension within a city, further experimentations as well as specifications like the materiality of the structure, are necessary.

To conclude, the proposed method enhances the argument that the exploration of natural evolutionary processes and ecological relations can provide new possibilities to architectural design. The bottom-up approach that was followed during this research resulted to the emergence of a self-organized and adaptive system that can potentially extend the existing space providing answers not only to the need of low-cost and temporary accommodations but also to the problem of nonexistent available space. In this context, the suggested method facilitates architects to form a mechanism for exploiting every possibility that space offers.

Appendix I

Rapid Prototyping

Rapid prototyping has been used as a starting point for examining the capacity of the structure for further fabrication through the translation of a generated computer model into a physical prototype (fig. 45). The three-dimensional nature of the physical prototype allows for closer examination and critical observation of the resulted form as it presents a scale, volume and materiality.

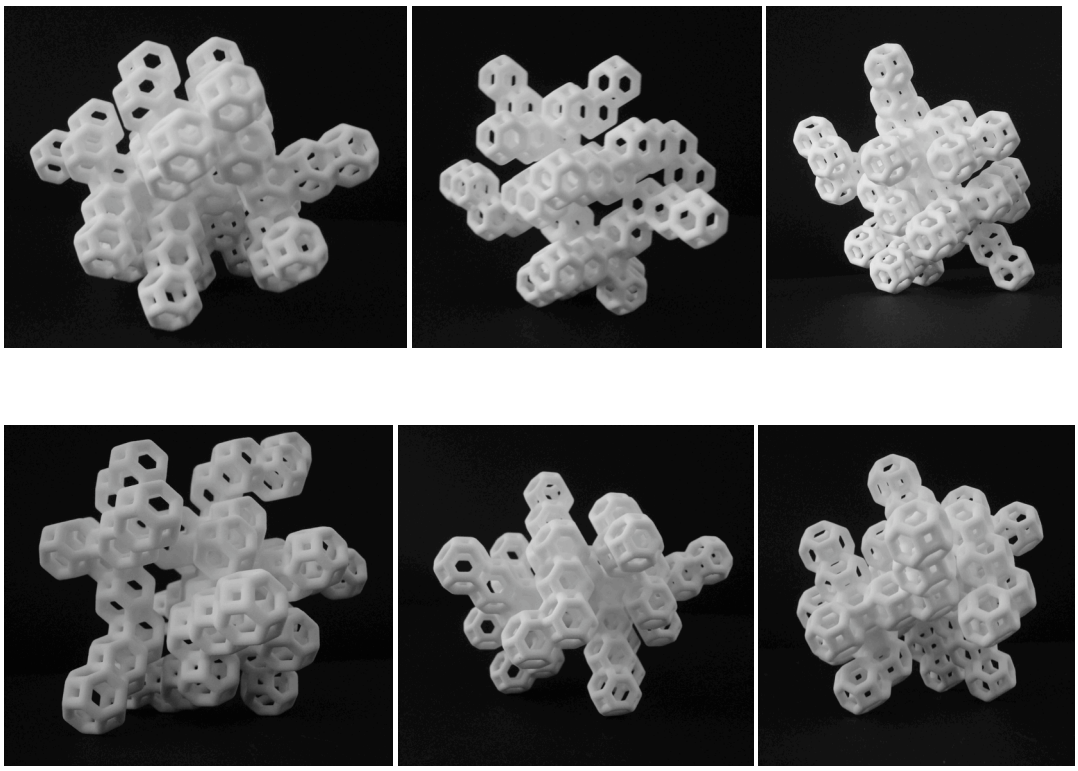


Figure 45: Rapid Prototyping models

The complexity of the form exhibits great interest and a contented analogy among the different diameters applied to the tubes that form the truncated octahedrons. The success in the manufacturing performance potentially offers the possibility for further fabrication in 1:1 scale.

Appendix II

Java Source Code – Basic Functions

In Main class:

```
static final float EDGE_LENGTH = 10.0f;
static final float STICKINESS = 1.0f;                //propability to stick
static final float STICKINESS_SHAPES = 1.0f;         //propability to stick parallel on x-y-z axis
static final float SIZE_FACTOR = 0.15f;              //to be determined based on weight of structure

//calculate radius of polyhedron
static final float circumRadius = 0.5f *sqrt(10) * EDGE_LENGTH;
static final float inRadius = 0.45f *sqrt(10) * EDGE_LENGTH;
static final float hexRadius = 0.5f *sqrt(6) * EDGE_LENGTH;
static final float sqRadius = sqrt(2) * EDGE_LENGTH;
static final float PARTICLE_RADIUS = circumRadius *2f;

int tubesSerial = 0;    // serial # for ID
int pointsSerial = 0;

// Mass calc temps
float totalMass;
PVector centerMass = new PVector(0,0,0);
float totalMassK;

// particles and Polyhedrons
ArrayList<DLAParticle> points = new ArrayList<DLAParticle>();
ArrayList<Shape> shapes = new ArrayList<Shape>();

ArrayList<Boxes> boxes = new ArrayList<Boxes>(); // constrain check
Wanderer wanderer; //walker
float currentRadius;



public void setup() {
        size(1200, 600, P3D);
        currentRadius = PARTICLE_RADIUS *2f;
        makeConstrains(); // build constrains
        initDLA();
}

public void draw() {
        background(255);

        for (int i = 0; i < 1000; i++) {
                wanderer.wander(currentRadius);
                if(checkCollisions(wanderer)) {
                        wanderer = new Wanderer( pointWithinSphere(currentRadius), this);
                }
        }
        for( int i=0; i < shapes.size(); i++ ) {
                pushMatrix();
                translate(shapes.get(i).centerPart.pos.x, shapes.get(i).centerPart.pos.y, shapes.get(i).centerPart.pos.z);
                renderScene(shapes.get(i));
                popMatrix();
        }
}
```

```
public void initDLA() {
	points.clear();
	points.add(new DLAParticle(new PVector(0,0,0),pointsSerial)); // initial seed
	pointsSerial++;
	shapes.add(new Shape(null,points.get(points.size()-1),tubesSerial,this));
	shapes.get(shapes.size()-1).style = 1;
	tubesSerial++;
	wanderer = new Wanderer(pointWithinSphere(currentRadius), this);
}

// add new particles
public boolean checkCollisions(Wanderer w) {

	for( int i=0; i < points.size(); i++ ) {
		if (w.v.dist(points.get(i).pos) > hexRadius*2f
		    && w.v.dist(points.get(i).pos) < hexRadius*2f + 0.1f
		    && random(1) < STICKINESS) {

			// position constrains and world constrains
			if( (meetConstrains(w.v, points.get(i).pos, PARTICLE_RADIUS ))
			    && checkCollision(w.v.x, w.v.y, w.v.z))    {

				for(int j=0; j < points.size(); j++ ) {
					if( !(points.get(j).pos == points.get(i).pos)
					    && !meetC(w.v, points.get(j).pos, PARTICLE_RADIUS) ) {
						return false;
					}
				}

				points.add(new DLAParticle(points.get(i), new PVector(w.v.x, w.v.y, w.v.z), pointsSerial));
				pointsSerial++;

				for(int j=0; j<shapes.size(); j++) {
					if (points.get(i).pos == shapes.get(j).centerPart.pos) {
						shapes.add(new Shape(shapes.get(j),points.get(points.size()-1),tubesSerial,this));
						shapes.get(j).shapeChildren.add(shapes.get(shapes.size()-1));

						if( calcCenterMassZZ() ) {     // mass calc, particle can stay or be killed
							calcParticleSizes(shapes.get(shapes.size()-1));
							tubesSerial++;
							break;
						} else {
							shapes.remove(shapes.size()-1);
							shapes.get(j).shapeChildren.remove(shapes.get(j).shapeChildren.size()-1);
							points.remove(points.size()-1);
							return false;
						}
					}
				}
			}

			float distFromOrigin = w.v.dist(new PVector(points.get(0).pos.x,
			points.get(0).pos.y,
			points.get(0).pos.z));

			// look for position farther away
			if (distFromOrigin > currentRadius - PARTICLE_RADIUS) {
				currentRadius = distFromOrigin + PARTICLE_RADIUS;
			}
			return true; // point added
		}
	}
	return false; // point not added
}
```

```
// increase weight of shapes
public void calcParticleSizes(Shape t) {
	float curWeight = SIZE_FACTOR;
	if (curWeight > t.shapeSize) {
		t.shapeSize = curWeight;
	}
	while (t.parent != null) {
		t = t.parent;
		curWeight += SIZE_FACTOR;
		if (curWeight > t.shapeSize) {
			t.shapeSize = curWeight;
		}
	}
}

// check for available space
public boolean meetC(PVector w, PVector p, float pRadius) {
	PVector c = new PVector(w.x,w.y,w.z);
	if(c.dist(p)>2*circumRadius  ) return true;
	return false;
}

public boolean meetConstrains(PVector w, PVector p, float pRadius) {
	PVector c= new PVector(w.x,w.y,w.z);
	float dist = 0.1f;
	float a = pow(sqRadius,2);
	float b = pow(hexRadius+hexRadius,2);
	float xz = sqrt(b-a);

	if (checkCollision(w.x, w.y, w.z)  && checkCollision(p.x, p.y, p.z)) {
		if(c.dist(p)>2*hexRadius && c.dist(p)<(2*hexRadius)+dist) {
			if(w.z<p.z) {
				if(w.x>p.x && w.y<p.y) w.set(p.x+sqRadius, p.y-sqRadius, p.z-sqRadius);
				if(w.x>p.x && w.y>p.y) w.set(p.x+sqRadius, p.y+sqRadius, p.z-sqRadius);
				if(w.x<p.x && w.y>p.y) w.set(p.x-sqRadius, p.y+sqRadius, p.z-sqRadius);
				if(w.x<p.x && w.y<p.y) w.set(p.x-sqRadius, p.y-sqRadius, p.z-sqRadius);
			}

			if(w.z>p.z) {
				if(w.x>p.x && w.y<p.y) w.set(p.x+sqRadius, p.y-sqRadius, p.z+sqRadius);
				if(w.x>p.x && w.y>p.y) w.set(p.x+sqRadius, p.y+sqRadius, p.z+sqRadius);
				if(w.x<p.x && w.y>p.y) w.set(p.x-sqRadius, p.y+sqRadius, p.z+sqRadius);
				if(w.x<p.x && w.y<p.y) w.set(p.x-sqRadius, p.y-sqRadius, p.z+sqRadius);
			}
			return true;
		}

		if(abs(w.x-p.x)>2*sqRadius && abs(w.x-p.x)<(2*sqRadius)+dist && random(1)<STICKINESS_SHAPES) {
			if(w.x>p.x) w.set(p.x+(2*sqRadius), p.y, p.z);
			if(w.x<p.x) w.set(p.x-(2*sqRadius), p.y, p.z);
			return true;
		}

		if(abs(w.y-p.y)>2*sqRadius && abs(w.y-p.y)<(2*sqRadius)+dist && random(1)<STICKINESS_SHAPES) {
			if(w.y>p.y) w.set(p.x, p.y+(2*sqRadius), p.z);
			if(w.y<p.y) w.set(p.x, p.y-(2*sqRadius), p.z);
			return true;
		}

		if(abs(w.z-p.z)>2*sqRadius && abs(w.z-p.z)<(2*sqRadius)+dist && random(1)<STICKINESS_SHAPES) {
			if(w.z>p.z) w.set(p.x, p.y, p.z+(2*sqRadius));
			if(w.z<p.z) w.set(p.x, p.y, p.z-(2*sqRadius));
			return true;
		}
	}
	return false;
}
```

```
public PVector pointWithinSphere(float radius) {
	PVector n = new PVector(points.get(0).pos.x, points.get(0).pos.y, points.get(0).pos.z);
	float z = random(-radius, radius);
	float radius_sq = radius * radius;
	float zsq = z * z;
	float theta = random(0, TWO_PI);
	float x = sqrt(radius_sq - zsq) * cos(theta);
	float y = sqrt(radius_sq - zsq) * sin(theta);
	PVector v = new PVector(x, y, z);
	n.add(v);
	return n;
}


// mass and moment calc
public boolean calcCenterMassZZ() {
	int serial = shapes.get(shapes.size()-1).serial_number;
	int shape = 0;
	float moment = 0f;
	float thress = 50f;

	// go down the branch one by one
	while (totalMass >= 0f) {
		for (int i = 0; i < shapes.size(); i++) {
			shapes.get(i).isCalc = false;
		}
		centerMass= new PVector(0,0,0);
		totalMass = 0f;

		for(int i=0; i<shapes.size();i++){
			if(serial == shapes.get(i).serial_number) {
				shape = i;
				break;
			}
		}

		if(shapes.get(shape).shapeChildren.size()>1) thress *=5f;

		shapes.get(shape).getCM(this); // recursive to children

		centerMass.div(totalMass);
		float dist = pow((centerMass.x-shapes.get(tube).centerPart.pos.x), 2)
			+ pow((centerMass.z-shapes.get(tube).centerPart.pos.z), 2);
		dist = sqrt(dist);
		dist = dist/10f;
		moment = dist * totalMass;
		if(moment > thress ) totalMass = -100f; // if over the limit set negative

		if( totalMass < 0f ) {
			break;
		} else {
			if(shapes.get(tube).parent!=null) {
				serial = shapes.get(tube).parent.serial_number;
			} else {
				break;
			}
		}
	}

	if(totalMass < 0f) return false;
	if(totalMassK!=0 && moment-totalMassK>6f) shapes.get(shapes.size()-1).style = 2;

	totalMassK = moment;
	return true;
}
```

```
// make world objects
public void makeConstrains() {
        float distSurf = sqRadius;

        //basic surface
        boxes.add(new Boxes(0,distSurf+5.1f,0,width,10f,500f) );
        //surrounding buildings
        boxes.add(new Boxes(-50,distSurf- 20f,+20f,80f,40f,150f) );
        {…}
}

public boolean checkCollision(float x, float y, float z) {
        for (int i = 0; i < boxes.size(); i++) {
                if(!(pointNotInside(boxes.get(i), x, y, z))) return false;
        }
        return true;
}

public boolean pointNotInside(Boxes l, float x, float y, float z){
        if( (x<(l.x+l.sX/2f+sqRadius)) && (x>(l.x-l.sX/2f-sqRadius))
           && (y<(l.y+l.sY/2f+sqRadius)) && (y>(l.y-l.sY/2f-sqRadius))
           && (z<(l.z+l.sZ/2f+sqRadius)) && (z>(l.z-l.sZ/2f-sqRadius)) ) {
                return false;
        }
        return true;
}

public void renderScene(Shape t) {
        float w = t.shapeSize+1.0f;
        float ramt = TWO_PI/3.0f;
        //create the vertices of the truncated octahedron
        PVector p0 = new PVector(t.shapeVertices[0][0],t.shapeVertices[0][1],t.shapeVertices[0][2]);
        PVector p1 = new PVector(t.shapeVertices[1][0],t.shapeVertices[1][1],t.shapeVertices[1][2]);
        {…}

        if( t.style == 1) {
                //draw faces of truncated octahedron
                beginShape();
                vertex(p0.x,p0.y,p0.z);
                vertex(p1.x,p1.y,p1.z);
                vertex(p2.x,p2.y,p2.z);
                vertex(p3.x,p3.y,p3.z);
                endShape(CLOSE);

                {…}
        }
        if( t.style == 2) {
                //draw 3D tubes
                renderCylinder(p0.x,p0.y,p0.z,p1.x,p1.y,p1.z,w,p0.dist(p1),ramt,w);
                renderCylinder(p1.x,p1.y,p1.z, p2.x,p2.y,p2.z,w,p1.dist(p2),ramt,w);
                {…}
        }
}

public void renderCylinder(float x1, float y1, float z1, float x2,float y2, float z2, float w, float h, float ramt, float w1) {
        float vx = x2-x1;
        float vy = y2-y1;
        float vz = z2-z1;
        if(vz == 0) vz = 0.00000001f;
        float v = sqrt( vx*vx + vy*vy + vz*vz );
        float ax = acos( vz/v );
        if ( vz < 0.0 )  ax = -ax;
        float rx = -vy*vz;
        float ry = vx*vz;
        pushMatrix();
        translate( x1, y1, z1);
        rotate3d(ax,rx,ry,0.0f);
        drawCylinder(w1, w,h, 10);
        popMatrix();
}
```

```
public void drawCylinder(float topRadius, float bottomRadius, float h, int sides) {
        float angle = 0;
        float angleIncrement = TWO_PI / sides;
        beginShape(QUAD_STRIP);
        for (int i = 0; i < sides + 1; ++i) {
                vertex(topRadius*cos(angle),topRadius*sin(angle), 0 );
                vertex(bottomRadius*cos(angle), bottomRadius*sin(angle),h );
                angle += angleIncrement;
        }
        endShape();
}

public void rotate3d(float angle, float _x, float _y, float _z) {
        float c = cos(angle);
        float s = sin(angle);
        float m = mag(_x, _y, _z);
        float x = _x / m;
        float y = _y / m;
        float z = _z / m;
        applyMatrix(  x * x * (1 - c) + c, x * y * (1 - c) - z * s, x * z * (1 - c) + y * s, 0,
                      y * x * (1 - c) + z * s, y * y * (1 - c) + c, y * z * (1 - c) - x * s, 0,
                      x * z * (1 - c) - y * s, y * z * (1 - c) + x * s, z * z * (1 - c) + c, 0,
                      0, 0, 0, 1);
}
```

In Shape class:

```
float shapeVolume;
float tubeVolume;
float shapeMass;
PVector shapeCenterMass;
private float density = 0.01f ;
float numEdges;
public DLAParticle centerPart;
float shapeSize;
public dla_Project d;
float dist1;
float dist2;
float shapeVertices [][];
int style;
public Shape parent;

ArrayList<Shape> shapeChildren = new ArrayList<Shape>();

public boolean isCalc;
PVector centerMass;
int serial_number;

public Shape(Shape _parent, DLAParticle centerPos, int s, dla_Project _d) {
        d = _d;
        centerPart = centerPos;
        centerMass= new PVector(0,0,0);
        shapeSize = 0.3f;
        serial_number = s;
        parent = _parent;
        numEdges = 36.0f;
        style = 1;
        dist1 = d.EDGE_LENGTH * PApplet.sqrt(2);
        dist2 = 0.5f*dist1;
        initializeShapeVertices(centerPos.pos);
}
```

```
public void initializeShapeVertices(PVector p) {
	shapeVertices = new float[][] { {dist2, 0, dist1}, {0, -dist2, dist1}, {-dist2, 0, dist1},
	                                {0, dist2, dist1}, {dist1, 0, dist2}, {dist1, -dist2, 0},
	                                {dist2, -dist1, 0}, {0, -dist1, dist2}, {-dist2, -dist1, 0},
	                                {-dist1, -dist2, 0}, {-dist1, 0, dist2}, {-dist1, dist2, 0},
	                                {-dist2, dist1, 0}, {0, dist1, dist2}, {dist2, dist1, 0},
	                                {dist1, dist2, 0}, {dist1, 0, -dist2}, {0, -dist1, -dist2},
	                                {-dist1, 0, -dist2}, {0, dist1, -dist2}, {0, -dist2, -dist1},
	                                {dist2, 0, -dist1}, {-dist2, 0, -dist1}, {0, dist2, -dist1} };
}

public float calcShapeVolume() {
	if(style == 1) {
		shapeVolume = 8f*PApplet.sqrt(2f)*PApplet.pow(d.EDGE_LENGTH,3);
		shapeVolume =shapeVolume/10f;
	}
	if(style == 2) {
		tubeVolume = PApplet.PI * d.EDGE_LENGTH * shapeSize * shapeSize;
		shapeVolume = tubeVolume * numEdges;
		shapeVolume =shapeVolume/10f;
	}
	return shapeVolume;
}

public float calcShapeMass() {
	shapeMass = calcShapeVolume() * density;
	return shapeMass;
}

public PVector calcShapeCenterMass() {
	float x = centerPart.pos.x;
	float y = centerPart.pos.y;
	float z = centerPart.pos.z;
	shapeCenterMass=new PVector(x,y,z);
	return shapeCenterMass;
}


public void getCM(dla_Project p) {
	PVector currentMass= new PVector(0,0,0);
	currentMass = calcShapeCenterMass();
	currentMass.mult(calcShapeMass());
	p.centerMass.add(currentMass);
	p.totalMass += calcShapeMass();

	for (int i = 0; i < shapeChildren.size(); i++) {
		if( !(shapeChildren.get(i).isCalc) ) {
			shapeChildren.get(i).getCM(p);
		}
	}
	isCalc = true;
}
```

In Particle class:

```
public PVector pos;
public DLAParticle parent = null;
ArrayList<DLAParticle> children = new ArrayList<DLAParticle>();
int serial_number;

public DLAParticle(PVector _pos, int s) {
	pos = _pos;
	children.clear();
	serial_number = s;
}
```

```
public DLAParticle(DLAParticle _parent, PVector _pos, int s) {
        parent = _parent;
        pos = _pos;
        children.clear();
        serial_number = s;
}
```

In Wanderer class:

```
public PVector v;
dla_Project parent;

public Wanderer(PVector _v, dla_Project p) {
        v = _v;
        parent = p;
}

// random motion, constrained by current radius
public void wander(float maxDistance) {
        PVector dir = new PVector (parent.random(-1,1),parent.random(-1,1),parent.random(-1,1));
        dir.normalize();
        dir.mult(parent.PARTICLE_RADIUS);
        v.add(dir);

        if (v.dist(new PVector(parent.points.get(0).pos.x,
                parent.points.get(0).pos.y, parent.points.get(0).pos.z)) > maxDistance + parent.PARTICLE_RADIUS) {

                // we're outside the sphere, change direction
                dir.mult(-1);
                v.add(dir);
        }
}
```

References

Addison, P.S. (1997), *Fractals and chaos: an illustrated course,* Institute of Physics Publishing, Bristol, UK.

Alexander, C. (1964), *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA.

Allen, J., Nio, I., van Oenen, G., Koekebakker, O. Ibelings, H., Acconci, V., Framis, A., van Lieshout, A., Ban, S., Horbelt, B. and Winter, W. (2003), *Parasite Paradise: A Manifesto for Temporary Architecture and Flexible Urbanism*, NAi Publishers, Rotterdam, Netherland.

Aranda, B. and Lasch, C. (2006), *Pamphlet Architecture 27: Tooling*, Princeton Architectural Press, New York.

Ball, P. (1999), *The Self-Made Tapestry: Pattern Formation in Nature*, Oxford University Press, New York, USA.

Bush, A. O., Fernández, J. C., Esch, G. W. and Seed, R. J. (2001), *Parasitism: The Diversity and Ecology of Animal Parasites*, Cambridge University Press, Cambridge, UK.

Carroll, G.C and Wicklow, D.T. (1992), *The Fungal Community: its organization and role in the ecosystem,* Marcel Dekker, New York, USA.

Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G. and Bonabeau, E. (2003), *Self-organization in Biological Systems*, Princeton University Press, New Jersey, USA.

Deverall, B.J. (1981), *Fungal Parasitism*, Edward Arnold, London, UK.

Gould, S.J. (2002), *The Structure of Evolutionary Theory*, Harvard University Press, Cambridge, MA.

Gow, N.A. and Gadd, G.M (1995), *The Growing Fungus*, Chapman & Hall, London, UK.

Jencks, C. (1987), *Modern Movements in Architecture*, Penguin, London, UK.

Jennings, D.H. and Lysek, G. (1996), *Fungal Biology: Understanding the fungal lifestyle*, BIOS Scientific Publishers, Oxford, UK.

Kaandorp, J.A. (1994), *Fractal Modelling: Growth and Form in Biology*, Springer-Verlag, Berlin, Germany.

Liang, S. and Kadanoff, L.P. (1985*), Phys.Rev.* A31, 2628.

Lindenmayer, A. (1968), *Mathematical models for cellular interactions in development*, Journal of Theoretical Biology, 18:280-299.

Mandelbrot, B.B (1983), *The fractal geometry of nature*, Freeman, San Francisco.

Meakin, P. (1983), *Phys. Rev.* A27, 2616.

Meakin, P., Ramanlal, P., Sander, L. M. & Ball, R.C. (1986), Phys. Rev. A34, 3325

Moore, D. (1998), *Fungal Morphogenesis*, Cambridge University Press, New York, USA.

Pearce, P. (1978), *Structure in Nature is a Strategy for Design*, MIT Press, Cambridge, MA.

Rácz, Z. and Vicsek, T. (1983), *Phys.Rev. Lett*. 51, 2382.

Ramanlal, R. and Sander, L.M. (1985), *Phys. Rev. Lett.* 54, 1828.

Steadman, P. (2008), *The Evolution of Designs: Biological Analogy in Architecture and the Applied Arts*, Routledge, New York, USA.

Thomas, F., Renaud, F. and Guegan, J.F. (2005), *Parasitism and ecosystems*, Oxford University Press, New York, USA.

Thompson, D.W. (1942), *On Growth and Form*, Cambridge University Press, Cambridge, UK.

Turing, A.M. (1952), *The chemical basis of morphogenesis*, Phil. Trans. Roy. Soc. Lond. B237:37-72.

Vicsek, T. (1992), *Fractal Growth Phenomena (Second Edition)*, World Scientific, London, UK.

Wigley, M. (1998), *Constant's New Babylon: the hyper-architecture of desire*, 010 Publishers, Rotterdam, Netherland.

Witten, T.A. and Sander, L.M. (1981), *Diffusion-limited aggregation, a critical phenomenon*, Phys. Rev. Lett. 47(19): 1400-1403.